Universidad Nacional Autónoma de México
Posgrado en Ciencia e Ingeniería de la Computación

Descubrimiento de objetos visuales sin supervisión usando palabras visuales co-ocurrentes

TESIS

Que para optar por el grado de:
Maestro en Ciencia e Ingeniería de la Computación

presenta:
David Abel Herrera Rosales

director de tesis:
Dr. Gibran Fuentes Pineda
IIMAS-UNAM

Ciudad Universitaria, CD. MX. Enero 2020

Resumen

En esta tesis hacemos minería de instancias visuales (MIV) usando un método llamado Sampled Min-Hashing (SMH). Se evalúa su escalabilidad y eficiencia con conjuntos de imágenes de distintos tamaños y también su comportamiento usando distintos parámetros del método. Hacemos una comparación entre el uso características manuales y características basadas en CNN, las características basadas en CNN es algo novedoso que no ha sido explorado en el problema de MIV. Analizamos la eficiencia de las instancias descubiertas, un análisis cualitativo de las instancias descubiertas y una comparación cuantitativa con las características manuales SIFT.

Agradecimientos

Al amor de mi vida, mi esposa Paulina que compartió conmigo esta travesía siempre alentándome.

A mis papas Nora y Abel, por los valores que me inculcaron, los ejemplos de perseverancia y sobre todo por su amor.

A mi hermana Erika, mi prima Cristi y su esposo Marco, por su apoyo y cariño por siempre.

A mis abuelos Felisa y Abel, Paula y Rodolfo, por su cariño y amor incondicional.

A mi tutor Gibrán por por su orientación y supervisión constante. Su apoyo fue invaluable para terminar esta tesis.

A mis amigos que hicieron mucho más placentera esta experienciay y a la UNAM, con el orgullo de ser universitario.

El Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas y el Posgrado en Ciencia e Ingenieria de la Computación por haberme brindado la oportunidad de continuar con mis estudios.

Al Consejo Nacional de Ciencia y Tecnologia por la beca proporcionada durante este posgrado.

Índice general

Índice de figuras

Índice de cuadros

Glosario

MIV	Minería de instancias visuales
BoW	*Bag-of-Words*
CNN	*Convolutional Neural Networks*
SMH	*Sampled Min Hashing*
SIFT	*Scale Invariant Feature Transform*
FIM	*Frequent Item Mining*
ToF	*Thread of Features*
GmH	*Geometric min-Hasing*
gLDA	*Geometric Latent Dirichlet Allocation*
DELF	*Deep Local Features*
LIFT	*Learned Invariant Feature Transform*
ANN	*Approximate Nearest Neighbors*
PP	Precisión promedio
PPM	Precisión promedio máxima
MPPM	Media de la precisión promedio máxima

Lista de símbolos

a	Un escalar
$\boldsymbol{a}$	A vector
$\boldsymbol{A}$	Una matriz
$\mathbf{A}$	Un tensor
$\mathbb{A}$	Un conjunto
$\{0, 1, \ldots, n\}$	El conjunto de todos los enteros entre 0 y n
a_i	Elemento i del vector $\boldsymbol{a}$, con indices comenzando en 1

1

Introducción

1.1. Motivación

Los avances tecnológicos como las cámaras digitales, las redes sociales y el almacenamiento en la nube han hecho que el contenido multimedia, en especifico imágenes, sean muy populares en Internet. Las imágenes son un medio para enriquecer contenidos en las paginas web, nos permiten compartir experiencias con amigos y familiares en redes sociales, y el almacenamiento en la nube nos permite siempre tenerlas accesibles desde cualquier dispositivo. Una plataforma para compartirlas es Instagram que contiene más de 40 mil millones y donde se suben más de 95 millones diariamente. Otra plataforma para almacenarlas es Google Photos donde se suben más de 1.2 mil millones diariamente. Existen muchos otros servicios de almacenamiento como son: Dropbox, Google Drive y One Drive, y servicios para compartir imágenes como son: Facebook, Snapchat.

La gran cantidad de imágenes en internet ha permitido a los investigadores crear bases de datos con millones de estas, como el conjunto de ImageNet (Deng et al., 2009) que contiene más de 14 millones agrupadas en más de 20 mil categorías. Desde el 2010 los concursos de ImageNet han fomentado el desarrollo de nuevos métodos escalables y robustos para los problemas de visión computacional como son: clasificación, detección, segmentación, descripción de imágenes y recuperación.

Es posible encontrar la aplicación de estos métodos en productos comerciales como Google Photos que permite reconocer y buscar personas y ubicaciones en fotos; otro ejemplo es Google Search, donde es posible buscar imágenes similares a una de muestra.

1.2. Planteamiento del Problema

Un problema especifico de visión computacional que requiere analizar grandes cantidades de datos es la minería de objetos (también llamadas instancias) visuales (MIV). MIV es el descubrimiento y extracción automática de instancias frecuentes dentro de una colección de imágenes. El termino de *instancia* se puede referir a objetos, sitios de interés, logos, personas, entre otras. Las instancias que descubre son especificas, por ejemplo, una botella de Coca-Cola es una instancia distinta a una botella de Mirinda. Este problema es distinto a otros tipos de problemas de visión computacional como: recuperación, donde se buscan las imágenes similares a una de consulta; clasificación, donde se busca categorizar en clases. El descubrimiento y extracción de instancias es esencial en muchas tareas de análisis de multimedia, algunos ejemplos son: arqueología de imágenes (Kennedy y Chang, 2008), donde se buscan las modificaciones a imágenes usando MIV como una medida de similitud entre 2 imágenes; predicción de tendencias, donde las instancias más frecuentes nos dan idea de tendencias; resumen en multimedia (Wang et al., 2012), en el que se genera un resumen visual de un contenido con las imágenes más importantes y anotación de imágenes (Wang et al., 2010), que es la asociación automática de textos a instancias.

Las instancias que se buscan en imágenes pueden ser muy pequeñas o muy grandes, pueden estar bien definidas o ser de mala calidad. Por otra parte, muchas imágenes no están etiquetadas y generarlas usualmente requiere intervención humana, lo cual puede ser muy laborioso y costoso, por ello es necesario contar con métodos que no requieran supervisión. La minería de instancias sigue siendo un problema abierto en visión computacional. Los métodos actuales sufren de problemas en la efectividad del descubrimiento de instancias pequeñas. Las instancias pequeñas pueden aparecer en áreas muy limitadas y difíciles de observar por un humano. Las variaciones como escala, rotación y oclusiones, requieren de métodos automatizados diseñados específicamente para instancias pequeñas y sólo pueden manejar pequeñas cantidades de datos. Otro problema es el escalamiento, una gran cantidad de imágenes trae consigo problemas de eficiencia y manejo de ruido que afecta los métodos diseñados para instancias pequeñas. En la actualidad existen pocos métodos que descubren instancias grandes y pequeñas en colecciones de imágenes de gran escala.

Tradicionalmente, los métodos de MIV se han basado en características locales diseñadas manualmente, como SIFT (Lowe, 2004), que nos permiten encontrar una similitud o correspondencia entre 2 imágenes. Una característica puede ser una sección especifica de una imagen como una puerta, ventana o domo. Este tipo de características se denominan puntos de interés y a menudo se describen por los píxeles que lo rodean. Otro tipo pueden ser bordes, como el contorno de un edificio, y se pueden describir por su orientación o apariencia local. Una ventaja de las características locales es que nos permiten encontrar similitudes a pesar de oclusiones y transformaciones, como cambios de escala o rotaciones. En el problema de MIV, usualmente se construye un vocabulario visual haciendo agrupamiento de todas la

características locales de una colección de imágenes. De esta forma, las imágenes se representan mediante un vector de frecuencias del vocabulario visual llamado modelo de bolsa de palabras (BoW, por las siglas en inglés de *Bag-of-Words*). Finalmente, para encontrar las instancias algunos métodos usan MinHash que es una técnica para estimar rápidamente que tan parecidos son 2 conjuntos o se puede construir un grafo basado en las similitudes de imágenes y usar un algoritmo de búsqueda en el grafo.

La figura 1.1 muestra la evolución del uso de características para recuperación de imágenes y las etapas donde dominaban los métodos basados en SIFT y redes neuronales convolucionales (CNN). En el año 2000 Smeulders et al. (2000) hacen un estudio de métodos usados hasta la fecha, posteriormente Sivic y Zisserman (2003) propusieron Video Google en 2003, donde introducen el modelo de BoW al problema de recuperación de imágenes y en el 2004 a la clasificación. Los métodos de agrupamiento K-Medias jerárquico y K-Medias aproximado, fueron propuestos por Nister y Stewenius (2006), y Philbin et al. (2007), respectivamente, estos métodos son mas escalables y por lo tanto permiten la construcción de vocabularios visuales grandes. En el 2008, Jegou et al. (2008) propuso Hamming Embeddings, donde se hace una representación binaria de las palabras visuales. Aunque los métodos basados en SIFT seguían avanzando, los métodos basados en CNN comenzaron a tomar más importancia. Siguiendo el trabajo pionero de Krizhevsky et al. (2012) quienes obtuvieron el estado del arte en clasificación en la competencia ILSRVC usando características basadas en CNN y superando los anteriores resultados por un amplio margen. Desde entonces, este tipo de características han reemplazado a las manuales en diversas tareas de visión computacional. En el 2014, Sharif Razavian

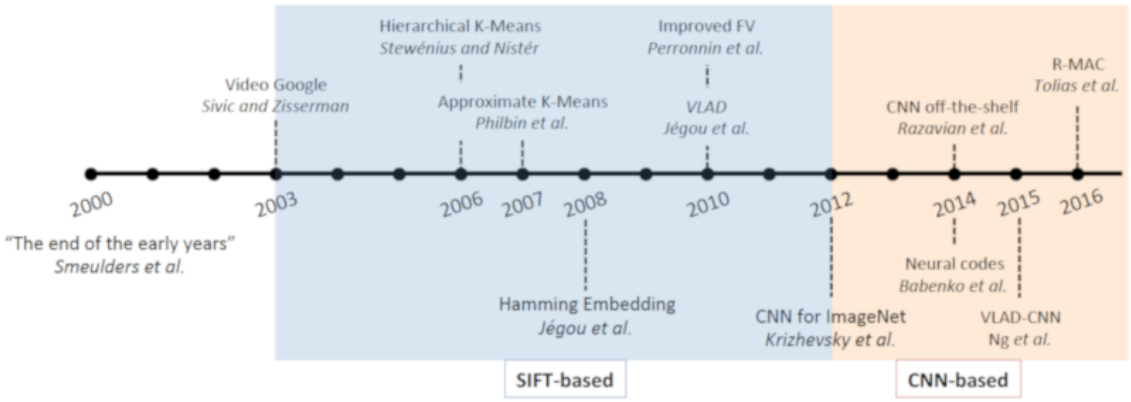

Figura 1.1: Acontecimientos importantes en recuperación de instancias. Imagen tomada de (Zheng et al., 2018).

et al. (2014) muestran que las características de una CNN entrenada en clasificación de objetos se pueden usar en una amplia gama de tareas como reconocimiento de escenas, reconocimiento de grano fino, detección de atributos y recuperación de imágenes. Babenko et al. (2014) fueron los primeros en ajustar un modelo CNN para la recuperación de instancias genéricas.

El uso de características locales basados en CNN no se ha aplicado al problema de MIV. En esta tesis, se experimenta y analiza un método de MIV usando distintas características basadas en CNN y en distintos conjuntos de imágenes de gran escala.

1.3. Metodología

Para analizar el impacto de características locales basadas en CNN y conjuntos de datos de gran escala en MIV se usará un método escalable para el descubrimiento de instancias grandes y pequeñas llamado Sampled Min Hashing (SMH) (Fuentes Pineda et al., 2011). El método de SMH aplica Min-Hashing al índice inverso para extraer conjuntos de palabras visuales co-ocurrentes. Posteriormente, aglomera de forma acumulativa las palabras co-occurente para descubrir las instancias.

Se analiza el comportamiento de SMH con distintas características locales en términos de exactitud en el descubrimiento de instancias y el uso de recursos computacionales, como memoria, CPU y tiempo de ejecución del algoritmo. Por otro lado, las características locales que se usan son SIFT y características basadas en CNN, en particular se usan los modelos de CNN de LIFT y DELF que representan el estado del arte.

Asimismo, algunos trabajos han mostrado que SMH deteriora su exactitud entre más grande sea la base de datos. En esta tesis se analizara como se afecta la exactitud y eficiencia al incrementar el número de imágenes por instancia.

1.4. Objetivos

El objetivo general de este trabajo de investigación es analizar el método SMH con bases de datos de gran escala, distintos tipos de características locales y parámetros de SMH. En particular nos enfocamos al descubrimiento de sitios de interés.

Objetivos específicos Esta tesis tiene los siguientes objetivos específicos:

- Analizar el comportamiento de SMH con descriptores SIFT con distintos tamaños de vocabularios, bases de datos y parámetros de SMH.
- Examinar distintos descriptores basados en CNN usados en recuperación y clasificación de sitios de interés.
- Seleccionar descriptores basados en CNN y evaluarlos con la técnica de SMH en el problema de descubrimiento de objetos de sitios de interés con distintos tamaños de vocabularios, bases de datos y parámetros de SMH.
- Evaluación cuantitativa del método propuesto y comparación con el uso de descriptores SIFT.

1.5. Estructura de la tesis

Esta tesis se encuentra organizada de la siguiente manera:

- **Capítulo 2**: Se describen los antecedentes de MIV donde se exponen brevemente los métodos más usados para descubrir instancias y el estado del arte. Asimismo, se exponen los antecedentes de los descriptores basados en CNN y el estado del arte.
- **Capítulo 3**: Se exponen los fundamentos de las técnicas usadas en la tesis como son SIFT, K-Medias, Min-Hash, ANN, CNN, DELF y LIFT.
- **Capítulo 4**: Se describe el uso del método de SMH con los distintos tipos de características locales.

- **Capítulo 5**: Se describen los métodos de evaluación cuantitativa, los conjuntos de datos usados y finalmente los resultados.

- **Capítulo 6**: Se resume las conclusiones y propone posibles direcciones de trabajo futuro.

2

Antecedentes

2.1. MIV

En esta sección hacemos una revisión breve de trabajos relacionados con MIV. Los trabajos que se presentan evalúan distintos tamaños de bases de datos y descubren instancias grandes y/o pequeñas. A continuación agrupamos los métodos con base en la cantidad de datos que analizan.

2.1.1. Colecciones de BD pequeñas

Uno de los primeros estudios sobre MIV lo modelaba como problema de minería de elementos frecuentes (FIM) (Quack et al., 2006), que fueron inicialmente utilizados para encontrar conjuntos de productos comprados juntos (por ejemplo, cerveza y pañales). Quack et al. analizan 1,500 cuadros de un vídeo donde se tratan porciones de una imagen como transacciones y palabras visuales como elementos. Aprovechan la propiedad de que en vídeos podemos identificar grupos de características que se trasladan constantemente de cuadro a cuadro debido a que las imágenes tienen una secuencia, esto permite mejorar la especificidad de las transacciones. Este método tiene problemas para identificar objetos no rígidos y texturas dispersas, ya que las primeras cambian la configuración espacial de las características y las segundas tiene pocas características.

Otra clase de métodos que abordan MIV se basan en el descubrimiento de tópicos (Sivic et al., 2005) como modelos de variables latentes por ejemplo: Hofmann (2017) y Blei et al. (2003). Los modelos de variables latentes representan cada imagen como una mezcla de tópicos, donde cada tópico corresponde a una sola clase de objeto. Estos métodos tienen la limitación de que el numero de tópicos se debe dar a priori y no son adecuados para grandes cantidades de datos, ya que lleva mucho tiempo inferir las distribuciones de los parámetros del modelo.

2.1.2. Colecciones de BD grandes

En (Philbin y Zisserman, 2008) se hace MIV en bases de datos de distintos tamaños, donde la mas grande tiene 1 millón de imágenes de Flickr. Las imágenes se representan como BoW, y se evalúan la similitud de cada imagen con toda la base de datos. Las coincidencias encontradas se validan a través de una verificación de consistencia espacial, usando una *affine homeography* para estimar las posiciones de las características. Se construyen un grafo de coincidencias usando la imágenes que tienen mayor coincidencia y los objetos se descubren particionando el grafo de correspondencias resultante en sub-grafos por medio de agrupamiento espectral. Este método es capaz de encontrar instancias grandes, sin embargo, la construcción del grafo y el agrupamiento espectral son muy ineficientes. En un trabajo posterior (Philbin et al., 2011) muestran que es posible disminuir el costo computacional de LDA aplicándolo a los sub-grafos, además incorporan la información geométrica al modelo de tópicos que llaman Geometric Latent Dirichlet Allocation (gLDA).

Se han desarrollado varios métodos más escalables para grandes bases de datos basados en Min-Hash. En (Chum et al., 2009a) se hace minería de imágenes similares

usando Min-Hash con la representación BoW de 100 mil imágenes de sitios de interés de Oxford. Las colisiones en las llaves son extraídas como objetos, a medida que el número de palabras visuales entre 2 imágenes disminuye, difícilmente se tratan como imágenes similares. En el caso de objetos pequeños, el número de palabras visuales es pequeño y difícilmente se encuentra una colisión con este método.

El método de GmH (Chum et al., 2009b) extiende Min-Hashing al considerar la relación geométrica entre las características locales. Evalúan el método en una base de datos de 100 mil imágenes de sitios de interés. En GmH se selecciona un conjunto de características en una imagen que tengan una palabra visual única dentro de la imagen y tengan al menos 3 características en su vecindario. Se genera una llave hash primaria con Min-Hash para este conjunto y se genera una llave secundaria con las características cercanas a la primera llave. Este método mejora la probabilidad de colisión para pequeñas instancias, gracias a la información de la llave hash secundaria.

El método de (Zhang et al., 2014) consiste en 2 pasos principales. El primero es la creación de Hilos de Características (ToF, por las siglas en ingles de *Thread of Features*) que consiste en un conjunto de características locales coherentes en múltiples imágenes. Se usan descriptores centrales, que son descriptores aumentados con los descriptores de sus vecinos. Dos descriptores centrales pertenecen a un hilo si comparten un número de palabras visuales. Posteriormente se hace agrupamiento de ToF usando Min-Hash, que permite extraer grupos de las colisiones en las tablas hash. Finalmente, los grupos de instancias se descubren con un esquema votación.

Li et al. (2015) consideran la estructura de conexión entre imágenes. Se extraen características que son aumentados usando las de sus vecinos. Se construye un grafo

donde los vértices son las imágenes y las aristas son las características similares entres las imágenes. Las instancias se descubren haciendo una búsqueda por profundidad en el grafo.

El método de SMH (Fuentes Pineda et al., 2011) también esta basado en Min-Hash, a diferencia del método de (Chum et al., 2009a), SMH aplica Min-Hashing al índice inverso para extraer conjuntos de palabras visuales co-ocurrentes. Posteriormente, agrupa de forma aglomerativa a las palabras co-ocurrentes que comparten muchas palabras visuales entre si. Los conjuntos extraídos se usan para generar los modelos de los objetos. El método de SMH descubre instancias grandes y pequeñas de manera eficiente y escalable. En el trabajo de ToF argumentan que el método de SMH solo funciona en conjuntos de datos de pequeña escala ya que el rendimiento cae rápidamente a medida que más imágenes están involucradas, ya que se introduce más ruido a la representación de palabras visuales.

En este trabajo se analiza el método de SMH a detalle, analizando su comportamiento usando distintos vocabularios visuales, conjuntos de datos y características visuales.

2.2. Características basados en CNN

Los métodos anteriores utilizan características locales manuales, aunque en el problema de MIV no se han explorado el uso de características basadas en CNN, si se ha explorado en otros tipos de problemas de visión computacional.

Los métodos para la extracción de características profundas basadas en CNN se pueden categorizar en tres de acuerdo con Zheng et al. (2018): CNN pre-entrenadas,

CNN ajustadas y CNN Híbridas. Las redes pre-entrenadas se modelan con un conjunto de datos distintos a la tarea especifica y las redes ajustadas son redes pre-entrenadas que se ajustan con datos parecidos a la tarea especifica. Los métodos híbridos usan secciones de la imagen que son pasados por una CNN múltiples veces para la extracción de características y su codificación e indexado son similares a SIFT. En algunos trabajos las características se extraen de los pesos de la capa completamente conectada (FC, por las siglas en ingles de *Fully Connected*) como (Zheng et al., 2016a; Sharif Razavian et al., 2014). Sin embargo, muchos métodos recientes usan las características de las capas convolucionales intermedias como (Zheng et al., 2016b; Yue-Hei Ng et al., 2015; Tolias et al., 2015) y han mostrado mejores resultados.

Una red CNN ajustada que extrae descriptores locales es DELF (Noh et al., 2017). El método de DELF es considerado el estado del arte para la búsqueda de imágenes similares (recuperación). Las características se extraen de de las últimas capas convolucionales. Se utiliza una ResNet50 (He et al., 2016) pre-entrenada con ImageNet, y ajustada usando un conjunto de datos de sitios de interés. Se utiliza un modelo de atención que consiste en 2 capas convolucionales para seleccionar un subconjunto de las características. Estos descriptores son más relevantes y nos ayudan mejorar la exactitud y usar menos recursos computacionales.

Una arquitectura CNN híbrida es LIFT (Yi et al., 2016) (por sus siglas en ingles de *Learned Invariant Feature Transform*). El método de LIFT consiste en tener 3 CNNs, una para la detección de puntos de interés, otra para la estimación de orientación y otra para la descripción de características. Se usan secciones de la imagen para entrenar cada una de las CNNs individualmente y se usan Transformadores

Espaciales (Jaderberg et al., 2015) para conectar los distintos componentes.. La arquitectura LIFT tiene muchas similitudes con SIFT, sin embargo es más lenta que DELF para obtener las características.

En este trabajo se exploran las características DELF y LIFT en el método de SMH.

3

Marco teórico

3.1. SIFT

Scale Invariant Feature Transform (SIFT) es un algoritmo que permite detectar y describir características locales de una imagen, las cuales son invariantes a la escala y rotación y relativamente robustas a transformaciones afines, cambios de punto de vista, adición de ruido y cambios en la iluminación. En esta sección damos una breve descripción del algoritmo de SIFT, para una explicación más detallada se remite al lector al artículo original de Lowe (2004).

Hay 4 pasos principales en el algoritmo de SIFT los cuales son:

1. Detección de extremos en la representación espacio-escala:

   Se buscan puntos de interés en distintas posiciones y escalas de la imagen que sean identificables con distintas vistas del mismo objeto. La detección de puntos de interés que sean invariantes a la escala se logra obteniendo características estables en distintas escalas usando una función continua de la escala llamado espacio-escala. Se usa la diferencia de gausianas (DG) para detectar puntos de interés estables en el espacio-escala. Las DG se obtienen calculando la diferencia del suavizado gaussiano de una imagen en distintas escalas σ, también llamado un octavo. Para generar el espacio-escala se obtiene un octavo con distintos tamaños de la imagen, es decir, se hacer un sub-muestreo

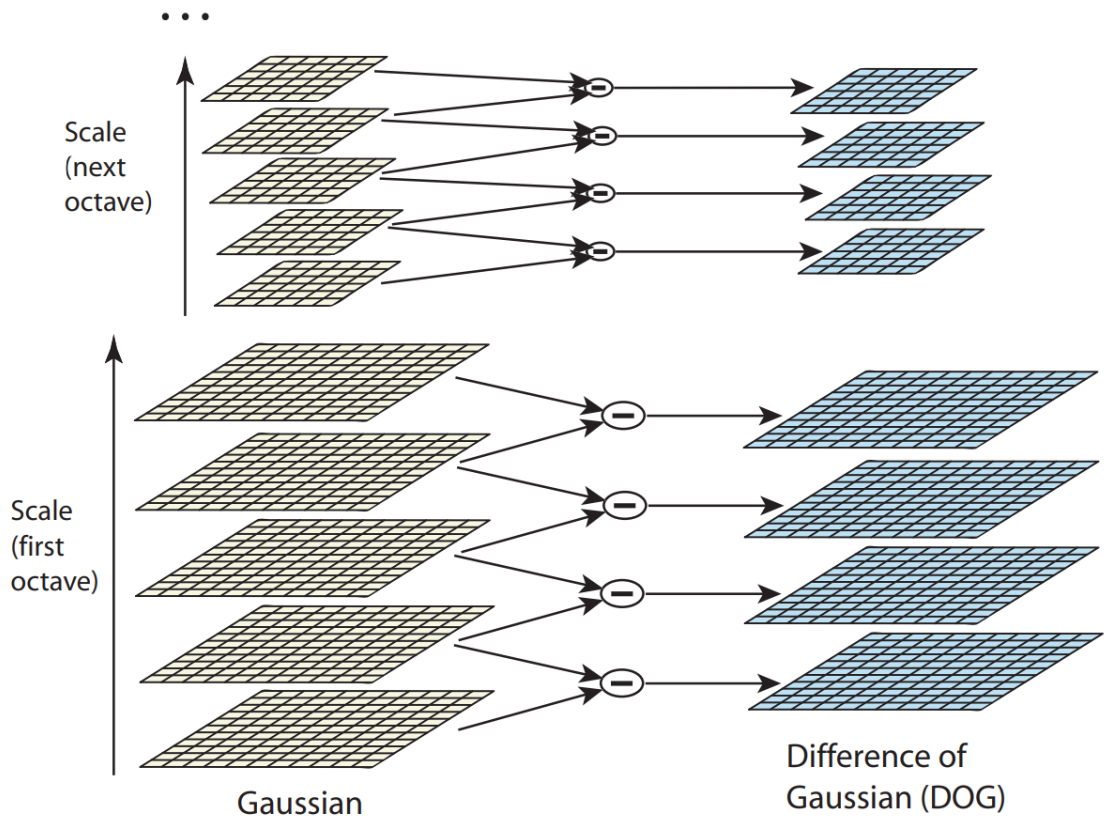

Figura 3.1: Piramide Gaussiana. Imagen tomada de (Lowe, 2004)

de la imagen y se obtienen las DG. La pirámide gaussiana en la figura 3.1 representa este proceso. Una vez que tenemos las DG, se buscan los extremos locales comparando un píxel con sus vecinos. Un píxel sólo se considera cómo un punto de interés potencial si es mayor o menor a estos vecinos. Por ejemplo, la figura 3.2 muestra cómo un píxel de una imagen es comparado con sus 8 vecinos además de 9 vecinos en la siguiente escala y 9 de la escala anterior.

2. Localización de puntos de interés:

   Se eliminan los puntos que tienen bajo contraste usando la expansión de Taylor en el espacio-escala. Los puntos de interés en los bordes tienen una mayor

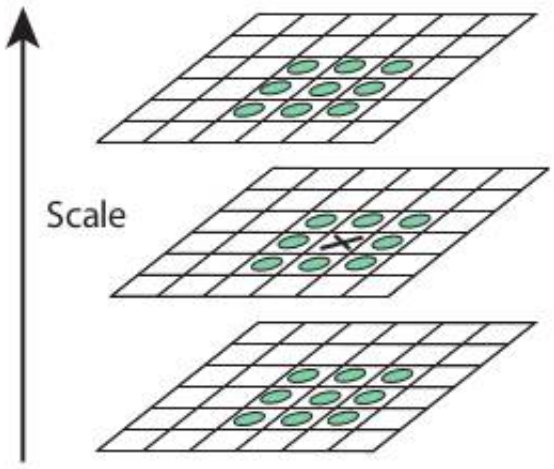

Figura 3.2: Comparación de píxeles en la representación espacio-escala. Imagen tomada de (Lowe, 2004)

respuesta en la DG y es necesario rechazarlos para tener una mayor estabilidad. Estos píxeles tienen una fuerte asimetría en la curvatura principal de la función DG y se usa una matriz Hessiana para calcularla. Se usa una proporción de la traza entre el determinante de la matriz Hessiana para identificar los bordes y rechazarlos.

3. Asignación de orientaciones:

   La invarianza a la rotación se logra representando cada punto de interés relativo a su orientación. Se obtiene una orientación dominante en cada punto de interés usando los píxeles cercanos y se calcula la magnitud del gradiente y la orientación de ellos. Posteriormente, se crea un histograma de orientaciones con 36 casillas que cubren 360 grados y se selecciona la orientación más grande en el histograma para asignarla al punto de interés.

4. Descriptor de puntos de interés:

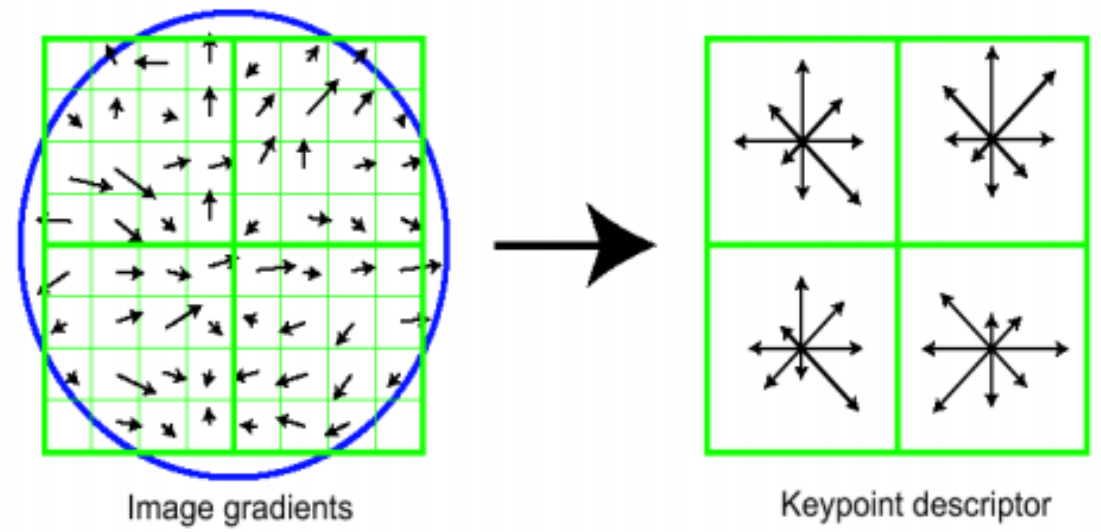

Figura 3.3: Descriptor SIFT. Imagen tomada de (Lowe, 2004)

Un punto de interés se describe primero obteniendo la magnitud del gradiente y las orientaciones de sus píxeles vecinos. Se consideran una región de 16×16 vecinos alrededor de cada punto de interés como se muestra a la izquierda de la figura 3.3, la figura muestra una región menor de 8×8. Se hace una rotación de las orientaciones relativas a la orientación dominante y estas muestras se acumulan en un histograma de orientaciones de 8 casillas que resume el contenido en 16 sub-bloques de 8×8 como se muestra a la derecha de la figura 3.3, la figura muestra una región menor de 4 sub-bloques de 4×4. La longitud de cada flecha corresponde a la suma de magnitudes del gradiente dentro de esta región. El descriptor se obtiene concatenando los 16 histogramas en un vector de 128 dimensiones.

3.2. Min-Hash

Min-Hash (Broder, 1997) es un algoritmo desarrollado para encontrar si dos documentos son "aproximadamente lo mismo" de manera eficiente. En esta sección damos una breve descripción de Min-Hash, para una explicación más detallada se remite al lector a las obra de Broder (1997) y Cohen et al. (2001).

La similitud Jaccard es un indicador de que tan parecidos son 2 conjuntos. La similitud de dos conjuntos, A y B, es un valor entre 0 y 1 de modo que cuando la similitud sea cercana a 1, es probable que los conjuntos sean muy parecidos. Su similitud Jaccard está dada por:

$$sim(A, B) = \frac{|A \cap B|}{|A \cup B|} \in [0, 1] \tag{3.1}$$

Un documento se puede representar como un conjunto de tokens, que pueden ser letras, palabras o líneas, por lo que es posible estimar que tan parecidos son dos documentos usando la similitud Jaccard. Supongamos que D_i y D_j son dos documentos cuyos tokens están contenidos en M. En Min-Hash, primero seleccionamos una permutación aleatoria π de los todos los tokens en M. Después de determinar π, el valor de min-hash para un documento D_i es su primer elemento después de que D_i se permuta de acuerdo con π. Es decir,

$$h(D_i) = min(\pi(D_i)) \tag{3.2}$$

En la práctica, la permutación aleatoria de los elementos se implementa mediante la asignación de un número aleatorio a cada token. El valor de min-hash de un

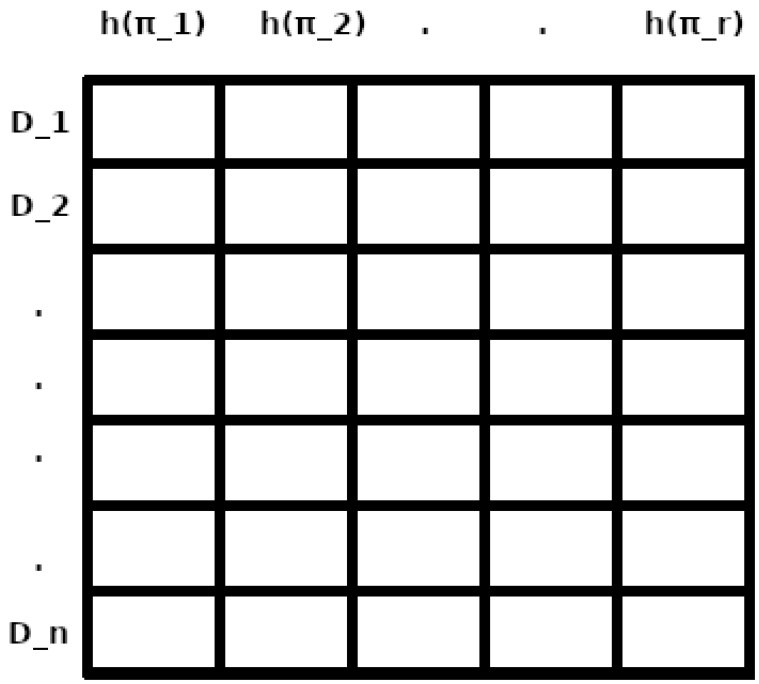

Figura 3.4: Representación de matriz de documentos y valores min-hash

documento se obtiene al encontrar el mínimo de los números asignados a sus tokens.

Es necesario obtener múltiples valores min-hash independientes para cada documento, por lo que usamos r permutaciones. El resultado se puede ver como una matriz, donde los renglones representan distintos documentos y las columnas son valores hash usando distintas permutaciones por columna, como lo muestra la figura 3.4. En particular, en Min-Hash la función hash de un documento regresa la concatenación de los valores min-hash de todo el renglón (las distintas permutaciones).

Se crean l matrices de la misma forma y se construye una tabla hash por cada matriz. Dos documentos entran en el mismo *bucket* de una tabla si su función hash $g_l(D_i)$ y $g_l(D_j)$ es la misma. La figura 3.5 muestra como un documento D_i

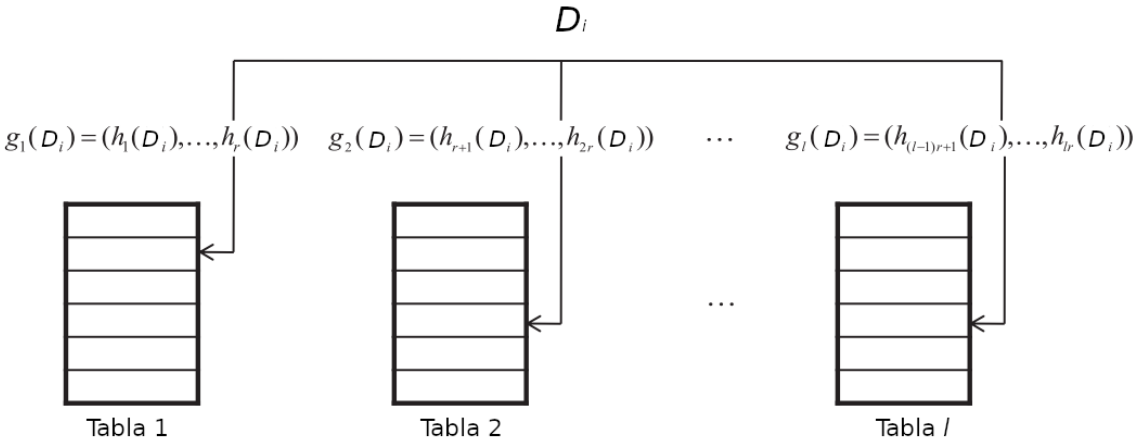

Figura 3.5: Construcción de tablas hash en Min-Hashing. Imagen tomada de (Fuentes Pineda et al., 2011) y editada

es agregado a sus *buckets* correspondientes en distintas tablas hash. Se espera que los documentos similares entren al mismo *bucket* en al menos una tabla hash. La probabilidad de que dos documentos D_i y D_j tengan el mismo valor hash para una función hash g_k esta expresada por:

$$P[g_k(D_i) = g_k(D_j))] = sim(D_i, D_j)^r \tag{3.3}$$

porque todos los valores de r min-hash en g_k tienen que ser los mismos. La probabilidad que D_i y D_j toman valores hash diferentes para todas las tablas l es $(1 - sim(D_i, D_j)^r)^l$, la probabilidad de que D_i y D_j se almacenen en el mismo hash bucket al menos en una tabla hash se expresa como

$$P_{colisión}[D_i, D_j] \approx 1 - (1 - sim(D_i, D_j)^r)^l \tag{3.4}$$

Es importante elegir correctamente r y l, ya que esto aumenta o disminuye la probabilidad de colisión.

3.3. K Medias

El agrupamiento K-medias es un tipo de aprendizaje no supervisado utilizado cuando se tienen datos sin etiquetar. El objetivo de este algoritmo es encontrar K grupos en los datos y funciona de manera iterativa para asignar cada dato a uno de los grupos K según sus características. Los datos se agrupan con base en su similitud usando una función de distancia. Los resultados del agrupamiento son los centroides de los agrupamientos K, que se pueden usar para etiquetar nuevos datos. Cada centroide es una colección de características que definen los grupos resultantes.

El algoritmo recibe como entrada el número de grupos K y el conjunto de datos. Se comienza con una estimación inicial para los centroides, que puede ser aleatoria y entonces se itera entre dos pasos:

- Asignación de datos a centroides: En este paso, cada dato se asigna a su centroide más cercano, en función una métrica de distancia. Formalmente, si c_i es la colección de centroides en el conjunto C, entonces cada dato x se asigna a un clúster basado en:

  $$\arg\min_{c_i \in C} dist(c_i, x)$$

  donde $dist()$ puede ser la distancia euclidiana estándar (L2).

- Actualización de centroides : En este paso se recalculan los centroides. Esto se hace tomando la media de los datos asignados al centroide.

El algoritmo itera entre los dos pasos hasta que se cumple un criterio de terminación (la suma de las distancias se minimice o se alcance un número máximo de

iteraciones). Este algoritmo converge a un resultado, pero puede ser un óptimo local, por lo que es necesario evaluar más de una ejecución del algoritmo con distintos centroides iniciales.

La complejidad en tiempo del algoritmo es:

$$O(n * K * I)$$

n : número de datos

K : número de agrupamientos

I : número de iteraciones

3.4. ANN

Un proceso muy costoso en visión computacional es la búsqueda de coincidencias más similares a vectores de alta dimensión, también llamado búsqueda de vecinos cercanos. No existen métodos que sean rápidos y exactos para obtener los vecinos cercanos por lo que es necesario usar soluciones inexactas para acelerar la búsqueda de forma sustancial.

Los métodos más populares de ANN (por sus siglas en inglés *Approximate Nearest Neighbors*) están basados en algoritmos de árboles como *kd-trees*, otras técnicas proyectan los datos a espacios de menor dimensionalidad como *locality-sensitive hashing* (LSH), otros construyen grafos donde cada vértice está asociado a un dato y sus vértices adyacentes son los vecinos cercanos y también se pueden usar métodos que cuantifican los vectores como *product quantization* (PQ) .

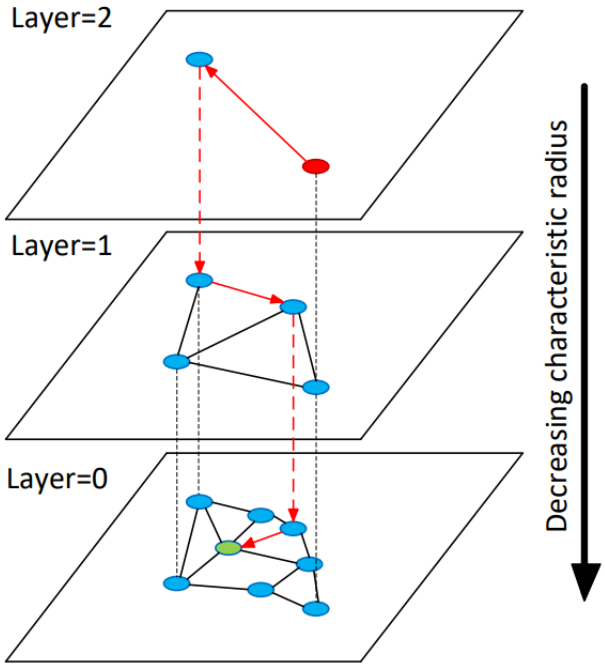

Figura 3.6: La búsqueda por capas en HNSW. Imagen tomada de Malkov y Yashunin (2018)

Un método que tiene muy buenos resultados en *benchmarks* de ANN (Aumüller et al., 2017) es HNSW (por sus siglas en ingles *Hierarchical Navigable Small World graphs*). HNSW (Malkov y Yashunin, 2018) es un algoritmo de grafos que utiliza una jerarquía de capas, donde la capa más alta contiene las aristas a vecinos lejanos y la capa más baja contiene las aristas a vecinos cercanos. La búsqueda de un vecino comienza por la capa más alta y desciende por las capas hasta encontrar el vecino en la capa más baja como se muestra en la figura 3.6 .

3.5. CNN

Las redes neuronales convolucionales (CNN por sus siglas en inglés) son redes neuronales especializadas basadas en la corteza visual del cerebro animal. En los últimos años, gracias al aumento del poder computacional, en particular los GPUs, la cantidad de datos de entrenamiento disponibles y los trucos para entrenamiento de redes profundas, las CNN han logrado lograr un rendimiento sobrehumano en algunas tareas de visión computacional.

Hubel y Wiesel (1959) proporcionaron ideas cruciales sobre la estructura de la corteza visual, en particular mostraron que las neuronas tienen un pequeño campo receptivo local, lo que significa que reaccionan solo a estímulos visuales ubicados en una región limitada del campo visual. Las CNN aprovechan la estructura espacial y local para procesar datos, en el caso de imágenes procesan datos de una matriz que contiene los valores de cada píxel. La representación de una imagen puede verse como una estructura tridimensional $w * h * d$ donde h es el alto, w es el ancho y d es la profundidad o el número de canales. Además, las CNN no están restringidas a percepción visual también tienen éxito en otras tareas como reconocimiento de voz o procesamiento del lenguaje natural.

Un acontecimiento importante es el articulo LeCun et al. (1989) donde introducen la arquitectura CNN llamada *LeNet-5* usada para reconocer números de cheques. Esta arquitectura presenta nuevos bloques de construcción esenciales es las CNN que son: capas convolucionales y capas de submuestreo que se describirán a continuación.

3.5.1. Capas convolucionales

El núcleo de una CNN son las capas convolucionales. La CNN está compuesta por múltiples capas convolucionales y cada capa tiene k filtros. Un filtro es una matriz cuyos valores son pesos que se aprenden durante el entrenamiento. El tamaño de un filtro es $r * r * d$ donde r es más pequeño que la dimensión de la imagen y d es igual al número de canales.

El nombre de las redes neuronales convolucionales tiene origen en el uso de una operación matemática llamada convolución. Las CNN son simplemente redes que utilizan la convolución en lugar de la multiplicación de matrices en alguna de sus capas. Una convolución, en el contexto de CNN, se entiende a grandes rasgos, como la acción de deslizar un filtro sobre una imagen. El filtro se desplaza a partir de la esquina superior izquierda de la entrada hasta la esquina inferior derecha, se mueve de izquierda a derecha y una vez que alcanza la esquina superior derecha, se mueve un elemento hacia abajo y de nuevo realiza un recorrido de izquierda a derecha. Este desplazamiento se puede ver en la figura 3.7. La convolución en CNN no corresponde precisamente a la definición de convolución usada en otras áreas, como matemáticas y análisis de señales, en el caso de señales es parecida a la correlación cruzada.

La convolución de una entrada con un filtro produce un mapa de características, es decir, el k-ésimo mapa de características en una capa está dado por la convolución de los pesos del k-ésimo filtro y los valores de entrada, más un desplazamiento llamado sesgo. Se puede pensar en las capas convolucionales como extractores de características de los datos de entrada. Zeiler y Fergus (2014) observaron que los filtros de las CNN son sensitivos a ciertos patrones, en capas iniciales estos patrones detectan elementos de bajo nivel como bordes, líneas y esquinas, mientras que las

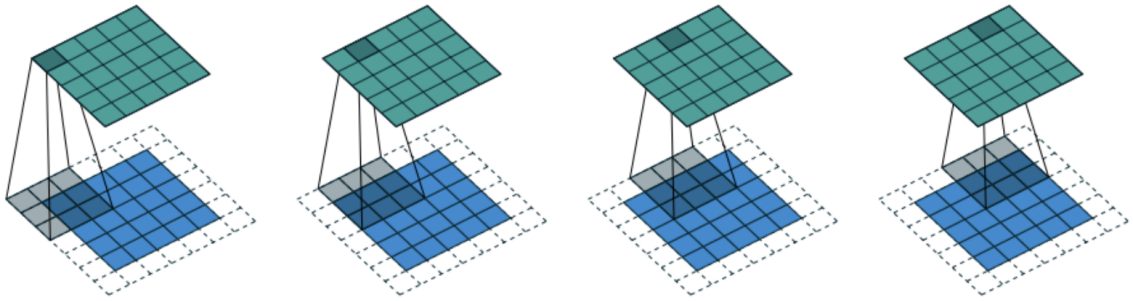

Figura 3.7: Desplazamiento de un filtro por una imagen. Imagen tomada de (Dumoulin y Visin, 2016)

capas de nivel superior extraen características cada ves más complejas a partir de las características de bajo nivel de la capa anterior. Los elementos de bajo nivel actúan como detectores de características manuales; sin embargo, los patrones de alto nivel son muy distintos a las características manuales.

Después de realizar la operación de convolución se aplica una función de activación. Existen muchas funciones de activación como: identidad, escalón, sigmoide y comúnmente se emplea la función ReLU.

Las capas convoluciones actualmente están optimizadas para correr en GPUs. Esto nos permite usar CNN profundas que consisten en muchas capas convolucionales. Adicionalmente, para entrenar una CNN profunda se requieren una gran cantidad de datos.

3.5.2. Capas de submuestreo

Las capas de submuestreo son comúnmente empleadas después de las capas convolucionales. Existen diversas maneras de hacer el muestreo, las más comunes son el muestreo máximo, el promedio y el global. Una capa de muestreo máximo regresa

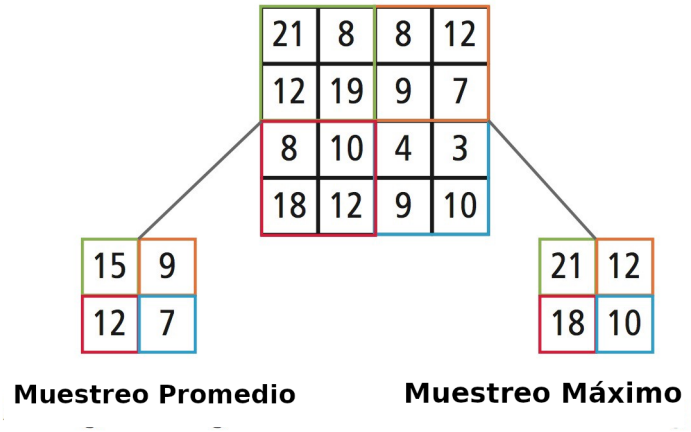

Figura 3.8: Ejemplo del proceso de muestreo promedio y máximo.

los valores máximos de regiones rectangulares de cierto tamaño determinado, mientras que una capa de muestreo promedio regresa el valor promedio de dicha región. Una capa de muestreo global reduce un mapa de características a un solo valor, el máximo o el promedio.

La figura 3.8 muestra el proceso de muestreo promedio y máximo de una entrada de tamaño $4*4$. Para el muestreo de $2*2$, la imagen se divide en cuatro matrices no superpuestas de tamaño $2*2$. En el caso del muestreo máximo se toma el máximo de los cuatro valores y en el caso del muestreo promedio y se toma el promedio de los 4 valores y se redondea al entero más cercano. Este tipo de capas ofrecen algunas ventajas como, por ejemplo, reducir la dimensionalidad de las características y hacerlas más manejables, reducir el número de parámetros y cálculos en la red. Además, hacen más robusta a la red neuronal a translaciones y rotaciones.

La figura 3.9 muestra el proceso del muestreo global promedio de 3 mapas de características de tamaños $6*6$, cada mapa de características se reduce a un solo valor, el promedio, por lo que tenemos una salida de $1*1*3$. Este tipo de capas tiene la ventaja de minimizar el sobre-ajuste mediante la reducción del número total

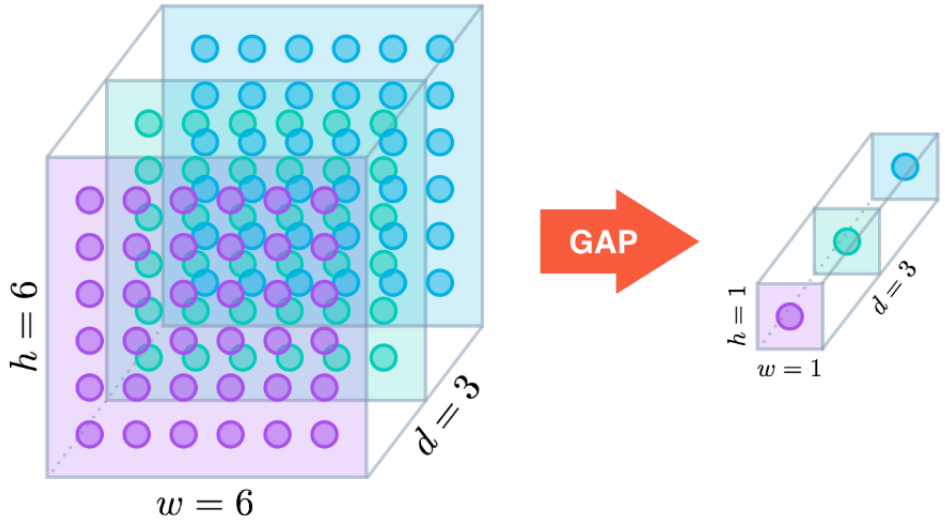

Figura 3.9: Ejemplo del proceso de muestreo global. Imagen tomada de (Cook, 2017)

de parámetros del modelo y facilita la localización de objetos (Zhou et al., 2016).

3.5.3. Arquitectura ResNet-50

Existen varios modelos de CNN populares para extraer características como son: AlexNet (Krizhevsky et al., 2012), VGGNet (Simonyan y Zisserman, 2014), GoogleNet (Szegedy et al., 2015) y ResNet (He et al., 2016).

Una ResNet (por sus siglas en ingles *Residual Neural Networks*) (He et al., 2016) es una CNN que es muy usada en problemas de visión computacional. Este modelo fue el ganador de la competencia *ImageNet* en el 2015.

Las redes neuronales profundas son difíciles de entrenar por el problema del desvanecimiento del gradiente. El gradiente se retro-propaga a capas anteriores, y a medida que se propaga el gradiente se vuelve más pequeño. La ResNet introduce la idea de omitir o saltarse algunas capas, creando bloques residuales. Una motivación

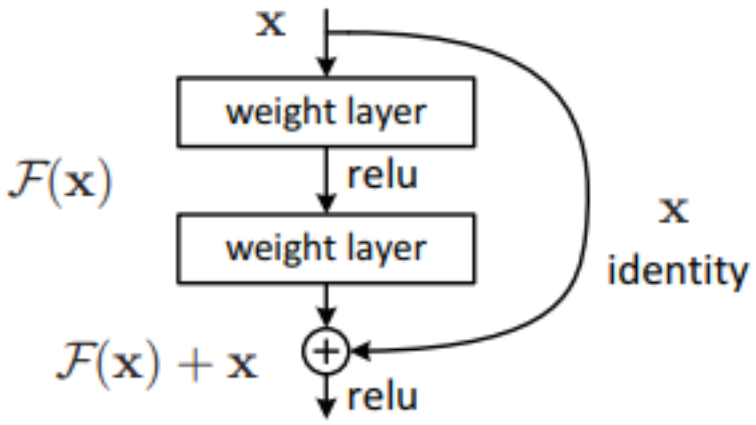

Figura 3.10: Bloque Residual

para saltarse capas es evitar el problema de la desaparición de los gradientes, reutilizando las activaciones de una capa anterior hasta que la capa adyacente aprenda su peso. La figura 3.10 muestra el bloque residual:

$$y = F(x, W_i) + W_s x$$

La entrada es x y la salida es y. La función $F(x, W_i)$ representa el mapeo residual que se quiere aprender y los autores mencionan que es más fácil optimizar esta función.

Existen distintas arquitecturas de ResNet como se puede ver en la figura 3.11 que muestra las arquitecturas de 18, 34, 50,101 y 152 capas.

3.5.4. Transferencia de conocimiento

La transferencia de conocimiento es una técnica popular en problemas de visión computacional donde los modelos pre-entrenados son usados como punto de partida

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Figura 3.11: Arquitecturas de ResNet. Imagen tomada de (He et al., 2016)

para la nueva tarea. Los modelos deben aprender millones de pesos y pueden tardar muchas semanas de entrenamiento, incluso con equipos de alto desempeño que emplean GPUs de última generación. Además requieren de una gran cantidad de datos para entrenarlos.

En la práctica, es muy poco común entrenar una CNN desde cero debido a que no se dispone de conjuntos de datos etiquetados de gran tamaño. En particular, el buen desempeño logrado por ResNet en ImageNet ha permitido emplear esta arquitectura para realizar transferencia de conocimiento.

La transferencia de conocimiento se puede realizar de las siguientes maneras:

1. Se ajustan los pesos de todas las capas convolucionales usando la base de datos de la tarea de interés.

2. Se ajustan los pesos de algunas capas convolucionales usando la base de datos de la tarea de interés. Por lo general se ajustan los pesos de las últimas capas convolucionaes, ya que las primeras capas detectan características generales

como como bordes, líneas y esquinas, y las últimas capas detectan estructuras más complejas dependientes del problema.

3. Se entrena únicamente la capa completamente conectada, sin embargo esté método no ha mostrado tan buenos resultados como los anteriores.

3.6. DELF

Deep Local Feature (DELF) es una red CNN ajustada que extrae características locales. El método de DELF (Noh et al., 2017) es considerado el estado del arte para la búsqueda de imágenes similares (recuperación).

Se utiliza una ResNet50 (He et al., 2016) pre-entrenada con ImageNet y se realizan 2 pasos adicionales: ajustar la red usando un conjunto de datos similares a la tarea especifica y entrenar un modelo de atención para seleccionar un subconjunto de los descriptores. A continuación describimos el proceso de ajuste y atención más en detalle.

3.6.1. Proceso de ajuste

El proceso de ajuste permite entrenar a la red para extraer características más adecuadas a la tarea específica. Se utilizan imágenes de un conjunto de sitios de interés y se pre-procesan de la siguiente manera:

- Centrar la imagen y recortarla en un cuadrado.
- Re-escalar la imagen a 250x250
- Recortar la imagen a un cuadrado aleatorio de 224x224

La red se entrena con una función de perdida de entropía cruzada para clasificar las imágenes.

3.6.2. Modelo de Atención

Se utiliza un modelo de atención para evaluar la relevancia de cada característica. El modelo de atención consiste en 2 capas convolucionales con una activación *softplus* y filtros de 1x1. Este modelo se entrena con un *dataset* de sitios de interés distinto al proceso anterior y se utiliza la misma función de perdida. Las imágenes se pre-procesan de la siguiente manera:

- Centrar la imagen y recortarla en un cuadrado.
- Re-escalar la imagen a 900x900
- Recortar la imagen a un cuadrado aleatorio de 720x720
- Escalar la imagen con $\gamma < 1$

La figura 3.12 muestra el proceso para ajustar la red y el entrenamientos del modelo de atención.

3.6.3. Reducción de dimensiones

Los descriptores se extraen de los mapas de características de las últimas capas convolucionales (conv4_x). Se extraen descriptores de una imagen en distintas escalas y se seleccionan las características que tengan un valor más alto en el modelo de atención.

Una vez que se tienen los descriptores seleccionados, se reducen las dimensiones de los descriptores a 40-dim usando PCA y se normalizan usando L_2.

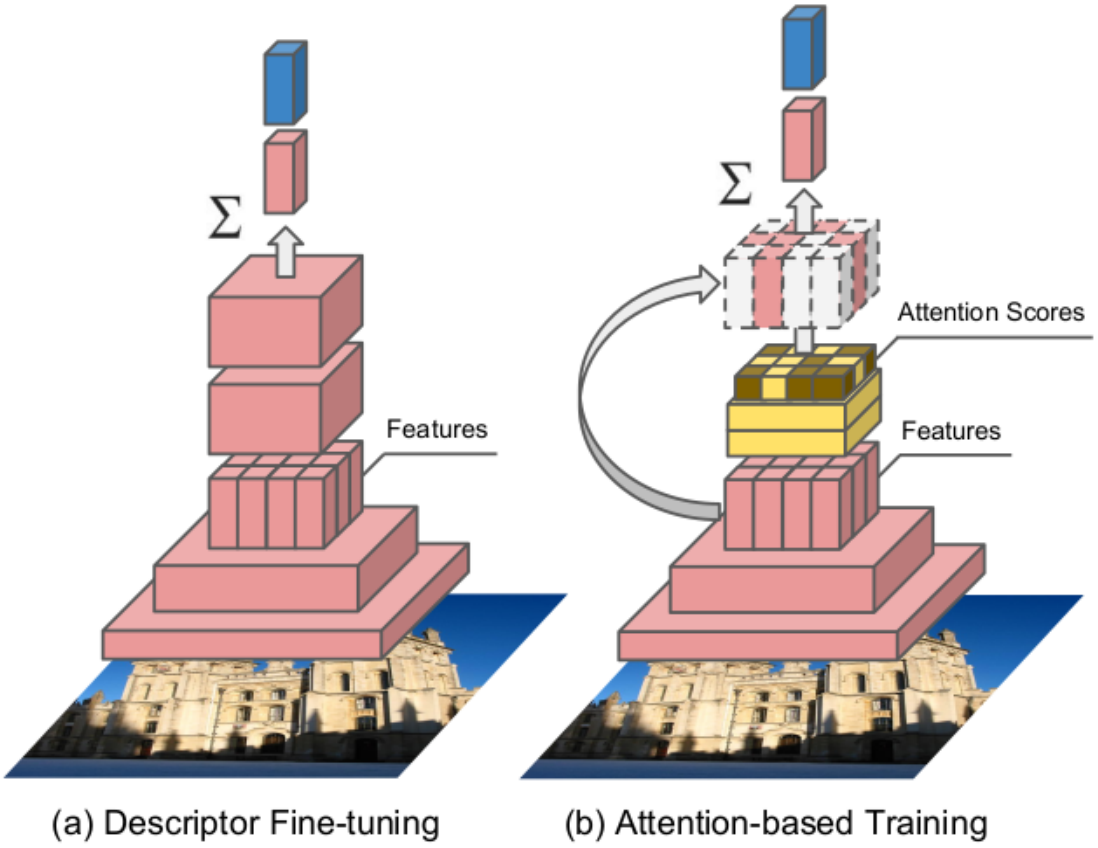

Figura 3.12: La arquitectura DELF. Imagen tomada de (Noh et al., 2017)

3.7. LIFT

Learned Invariant Feature (LIFT) (Yi et al., 2016) es una arquitectura que detecta puntos de interés, estima orientaciones y describe características. Tiene tres componentes que se alimentan entre sí: el detector, el estimador de orientaciones y el descriptor. Cada uno esta basado en CNN y utiliza Transformadores Espaciales (Jaderberg et al., 2015) para conectar los distintos componentes. Se construye una red Siamesa y la entrenan con características obtenidas de *Structure-from-Motion* (SfM) en distintos puntos de vista e iluminaciones. El entrenamiento se realiza con

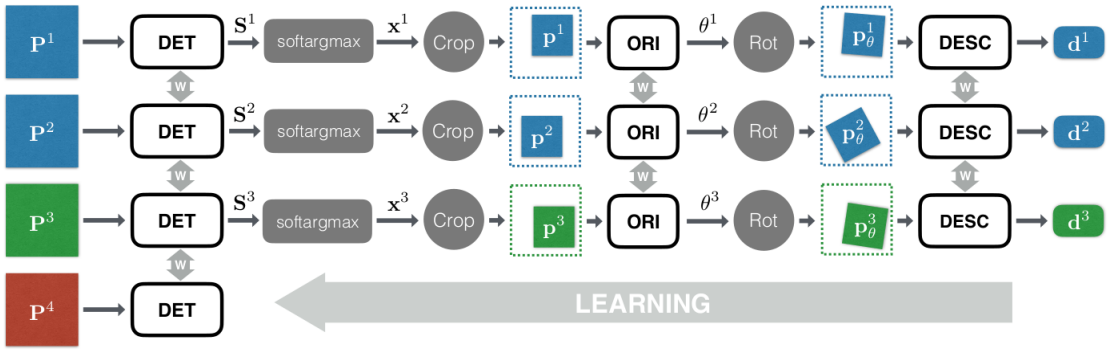

Figura 3.13: La arquitectura siamesa de LIFT. Imagen tomada de (Yi et al., 2016)

secciones de la imagen, primero se entrena el Descriptor que posteriormente es usado para entrenar el Estimador de Orientaciones y finalmente se entrena el Detector usando el Descriptor y el Estimador de Orientaciones.

Utilizaron un conjunto de datos de *Piccadilly Circus* de Londres y el Foro Romano para reconstruir el objeto 3D. Picadilly contiene 58 mil puntos de interés y el foro romano contiene 51 mil puntos de interés.

Se utiliza una red siamesa de cuatro ramas representada en la figura 3.13. El entrenamiento de la red se hace con secciones de una imagen lo suficientemente pequeñas para asegurar que sólo tengan un punto de interés, además se usan secciones que no tienen ningún punto de interés.

Usamos cuatro secciones de imagen, las secciones *P1* y *P2* corresponden al mismo punto de interés visto desde distintos puntos de vista, *P3* tiene un punto de interés distinto y *P4* no contiene ningún puntos de interés.

El descriptor usa una CNN con 2 capas seguida por unidades tagenciales hiperbolicas, submuestreo l2 y *local subtractive normalization*. La entrada es la posición y la orientación de la característica en la sección de imagen. Se entrena el descriptor con una función de perdida que minimiza la distancia entre el descriptor obtenido

y el esperado. En el caso de secciones de imagen que tienen un punto de interés distinto o no tienen punto de interés, se utiliza una función distinta para la perdida.

El estimador de orientaciones es una red que recibe como entrada una región que contiene un punto de interés obtenido por el SfM y busca predecir una orientación que minimiza las distancias entre los descriptores para diferentes vistas del mismo punto. Se utiliza el descriptor ya entrenado anteriormente para obtener los descriptores. En esta CNN no se utilizan imágenes que no tienen puntos de interés o tienen otros puntos de interés.

El detector es una red con una capa convolucional seguida de una activación *piecewise linear*, recibe como entrada una porción de imagen y obtiene un mapa de puntajes. El Descriptor y el Estimador de Orientación ya se entrenar en este punto, así que buscamos buscamos minimizar las distancias entre los descriptores para diferentes vistas del mismo punto y maximizar el puntaje de clasificación.

4

Descubrimiento de Objetos

En esta sección describimos el método usado para el descubrimiento de objetos que consiste en extraer características de los conjuntos de datos. Se crean vocabularios visuales de distintos tamaños con las características y posteriormente se representa cada imagen con estos vocabularios en una bolsa de palabras. Finalmente construimos un índice invertido al cual le aplicamos el método de SMH. A continuación describimos cada proceso a más detalle.

4.1. Modelo bolsa de palabras

El modelo BoW es una representación simplificada de un documento utilizada en procesamiento del lenguaje natural. Un documento se representa como la bolsa de sus palabras, sin tener en cuenta la gramática e incluso el orden de las palabras pero manteniendo la multiplicidad de las palabras. Este modelo fue inicialmente utilizado en imágenes por Sivic y Zisserman (2003) donde hacen una analogía visual de una palabra que se obtiene cuantificando las características locales. Este modelo a sido bastantes exitoso en tareas de reconocimiento y clasificación visual.

4.1.1. Extracción de características

Se extraen las características SIFT, DELF y LIFT de las imágenes. Las características SIFT son usadas comúnmente y representan nuestra comparación base.

Se utiliza el código de Perd'och et al. (2009)[1] para extraer las características SIFT, este código utiliza la pirámide gaussiana de Lowe (2004), obtiene extremos locales y descubre la forma local afín usando el método de Lindeberg (1998). Finalmente se obtiene el descriptor SIFT de Lowe (2004).

Usamos 2 tipos de características basadas en CNN que representan el estado del arte. La características DELF se obtienen de una CNN ajustada y usa un modelo de atención para obtener las más relevantes. Usamos el código proporcionado por Noh et al. (2017)[2]. Además, las características tienen menos dimensiones que SIFT. La extracción de características LIFT se hace con la implementación del CVLAB de Lausanne Federal Institute of Technology de Suiza (EPFL) que es una versión de Tensorflow del código original[3]. Este método pasa cada imagen por tres CNN y es lento al extraer las características.

4.1.2. Construcción del vocabulario visual

El objetivo de un vocabulario visual $V = \{v_1, .., .., v_N\}$ es cuantizar las características locales en agrupamientos que posteriormente representan las palabras visuales. En nuestro método usamos el agrupamiento K-Medias para obtener las características cuantizadas. El algoritmo de K-Medias busca las coincidencias más similares para agrupar las características, en nuestro caso las características son vectores de alta dimensión. La búsqueda de vecinos cercanos en altas dimensiones es muy costoso computacionalmente, sin embargo existen métodos aproximados para mejorar la eficiencia, nosotros elegimos utilizar el método de HNSW por sus buenos

[1] `https://github.com/perdoch/hesaff`
[2] `https://github.com/tensorflow/models/tree/master/research/delf`
[3] `https://github.com/cvlab-epfl/tf-lift`

resultados en otros experimentos (Aumüller et al., 2017) y usamos el código de [4].

El método de K-Medias recibe como entrada el tamaño del vocabulario visual $|V|$, el número de iteraciones y las características de las imágenes. Usamos 30 iteraciones y en cada una usamos el HNSW para encontrar el centroide más cercano a una característica. El método de HNSW acelera la búsqueda de centroides y la construcción del vocabulario visual de forma sustancial. En algunos experimentos no es posible guardar todas las características en memoria RAM por lo que hacemos un muestreo aleatorio.

Una imagen se puede representar en el modelo BoW como un vector de frecuencias de las palabras visuales $D_i = (t_1, t_2, ..., t_V)$. En nuestro método las representamos con BoW un binario, que indica la presencia o la ausencia de la palabra visual. Esto resulta una representación mas compacta con una discruminación buena para vocabularios visuales grandes. En Jégou et al. (2009) muestran que el BoW binario ligeramente mejora la calidad de búsqueda que en el BoW estándar.

Adicionalmente, haciendo una analogía con *stopwords* que son palabras que aparecen poco o muy frecuentes, nosotros quitamos las palabras visuales que ocurren en menos del 0.1 % y en más de 30 %.

4.2. Índice invertido

El índice invertido permite búsquedas rápidas en de palabras visuales en el BoW. Es fácil de desarrollar y es una estructura de datos popular utilizada en los sistemas de recuperación de documentos, utilizada a gran escala, por ejemplo en los motores de búsqueda. Se construye el índice invertido a partir del BoW y contiene para cada

[4] `https://github.com/nmslib/hnswlib`

palabra visual v_i una lista de las imágenes donde aparece v_i. Denotamos el conjunto de imágenes que contienen v_i por $\hat{v_i}$ y se refieren a él como el conjunto de ocurrencias de v_i.

4.3. SMH

El método de SMH nos permite descubrir objetos usando el índice invertido. Usamos Min-Hash para encontrar la similitud entre $\hat{v_i}$ y $\hat{v_j}$, $\hat{v_i}$ contiene el conjunto de imágenes que contienen v_i, por lo tanto la similitud Jaccard $sim(\hat{v_i}, \hat{v_j})$ es una medida de que tan frecuentemente $\hat{v_i}$ y $\hat{v_j}$ aparecen en las mismas imágenes.

El conjunto de palabras visuales que aparecen frecuentemente en las mismas imágenes colisionan en el mismo hash *bucket* y las llamamos palabras co-ocurrentes representadas por ϕ. Este conjunto de palabras co-ocurrentes ϕ aparecen juntas en las mismas imágenes y esperamos que pertenezcan al mismo objeto. En todos los experimentos evaluamos el comportamiento de SMH con distintos número de permutaciones y tablas hash que controlan la probabilidad de colisión. Un valor alto en el número de permutaciones r disminuye la probabilidad de colisión y un valor alto de tablas l aumenta la probabilidad de colisión, usamos la fórmula de Fuentes-Pineda y Meza-Ruiz (2019) para calcular l y tener una probabilidad de colisión del 50 %.

$$l = \frac{log(0.5)}{log(1 - \eta^r)} \tag{4.1}$$

Una vez que tenemos los conjuntos de palabras co-ocurrentes descartamos las palabras visuales que consideramos ruidosas. Para evaluar si una palabra visual es

ruidosa, obtenemos las imágenes donde aparece más de α veces, si el número de imágenes es menor a un umbral β entonces se considera ruidosa. Las palabras visuales ruidosas se eliminan de ϕ y si ϕ es menor a un umbral también lo descartamos.

El descubrimiento de objetos se hace agrupando palabras co-ocurrentes que comparte muchas palabras visuales. Dos conjuntos ϕ_i y ϕ_j se agrupa de forma aglomerativa si tienen un coeficiente de traslape $ovr > \epsilon$ que está dado por:

$$ovr(\phi_i, \phi_j) = \frac{|\phi_i \cap \phi_j|}{min(|\phi_i|, |\phi_j|)} \in [0, 1] \tag{4.2}$$

Usamos Min-Hash para encontrar que conjuntos pueden agruparse, ya que :

$$\frac{|\phi_i \cap \phi_j|}{min(|\phi_i|, |\phi_j|)} \geq \frac{|\phi_i \cap \phi_j|}{|\phi_i \cup \phi_j|} \tag{4.3}$$

Esperamos que conjunto de palabras co-ocurrente que comparten muchas palabras visuales entren en el mismo *bucket* hash.

Se construye un grafo donde cada conjunto de palabras co-ocurrentes ϕ_i es un nodo y existe una arista entre dos nodos si su $ovr > \epsilon$. Los vértices que pertenecen al mismo componente conectado se combinan en un agrupamiento que representa el objeto descubierto.

4.4. Recuperación de objetos descubiertos

El método obtiene los objetos descubiertos combinando los conjuntos de palabras co-ocurrentes que están representados por sus palabras visuales. Las imágenes están representadas por sus palabras visuales en el BoW, así que podemos determinar si una imagen pertenece al objeto por el número de palabras visuales que comparten.

Obtenemos las imágenes que comparten muchas palabras visuales con el objeto descubierto ya que es probable que contengan el objeto. Las imágenes recuperadas se ordenan de acuerdo con la cantidad de palabras visuales compartidas para mostrar las imágenes más relevantes primero.

5

Experimentación y evaluación

En esta sección describimos los distintos conjuntos de imágenes, explicamos las métricas usadas para evaluar los experimentos y discutimos los resultados obtenidos. Todos los experimentos se corrieron en una Intel(R) Core(TM) i7-6700K CPU 4.00GHz y para la extracción de características de CNN se uso una GPU Titan X con 64 GB de memoria.

5.1. Conjunto de imágenes

Existen conjuntos de imágenes comúnmente utilizados para la evaluación de las técnicas de minería de instancias. El Oxford5K (Philbin et al., 2007) es una de los más populares con 5,062 imágenes capturadas de 11 sitios de interés en Oxford de Flickr[1]. Algunos ejemplos de estas imágenes se pueden ver en la figura 5.1. El conjunto de imágenes también contiene la categoría de cada una de las imágenes. Adicionalmente, las imágenes son clasificadas con base en su calidad:

- *Good*: una imagen clara del objeto.
- *OK*: más del 25 % del objeto es claramente visible.
- *Junk*: menos del 25 % del objeto es visible, hay una gran cantidad de oclusiones o distorsiones.

[1] https://www.flickr.com/

Figura 5.1: Ejemplos de imágenes de Oxford5k

La desventaja de Oxford5K es su tamaño pequeño ya que en la actualidad existen conjuntos de imágenes con muchas más imágenes.

El conjuntos de imágenes de Google-Landmarks (Noh et al., 2017) es mucho más grande, contiene más sitios de interés y tiene más retos visuales. El conjunto de imágenes contiene 1,060,709 imágenes de 12,094 sitios de interés. Algunos ejemplos de estas imágenes se pueden ver en la figura 5.2.

Las imágenes se recopilaron de información pública y cada imagen tiene asociada una coordenada GPS, si la distancia entre la imagen y un sitio de interés es menor a un umbral entonces la imagen se considera parte del sitio de interés. Es importante notar que los datos sólo contiene la dirección URL de las imágenes y es posible que algunas imágenes ya no estén disponibles. Además el conjunto de imágenes esta desbalanceado, por ejemplo muy pocos sitios de interés tienen más de 3 mil imágenes

Figura 5.2: Ejemplos de imágenes de Google-Landmarks

y hay muchos sitios con solo 1 imagen. En el cuadro 5.1 mostramos la distribución de imágenes para los 20 sitios de interés con más imágenes.

5.2. Métricas

Para evaluar desempeño de nuestro método usamos la metodología de evaluación empleada por Fuentes Pineda et al. (2011) y Philbin y Zisserman (2008). El método de SMH descubre múltiples objetos pero como tenemos una tarea no supervisada, los objetos no están explícitamente asignados. Se evalúan todos los objetos con todos los sitios de interés, y a cada sitio de interés se asigna el objeto que comparte más imágenes.

Para cada objeto obtenemos la precisión promedio (PP) con respecto al sitio de interés que tiene un rango de 0 a 1 y está dada por el área bajo la curva de

Sitio de interés	Número de imágenes
Plaza de San Pedro	49,232
Coliseo	49,078
Ciudad de Chicago	23,035
Alhambra	18,062
Puente Carlos	13,015
Torres Petronas	10,924
Piazzale Michelangelo	9,372
MNAC	9,069
Puente de Rialto	8,951
Panteón de Agripa	8,872
Isla de Alcatraz	8,853
Palacio Imperial de Hofburg	8,525
Catedral de Berlín	7,654
Frankfurt	6,799
Santa Sofia	6,462
Palazzo Comunale	6,321
Casa Batlló	5,260
Hofkirche	5,223
Half Dome	5,183
Lagoa	4,845

Cuadro 5.1: Distribución de imágenes en los 20 sitios de interés con más imágenes.

precisión-exhaustividad (ROC). La precisión es el número de imágenes positivas entre el número total de imágenes evaluadas y la exhaustividad es el número de imágenes evaluadas positivas entre el número total de imágenes positivas. De todos los objetos descubiertos se toma el objeto con la máxima PP (precisión promedio máximo o PPM) para cada uno de los sitios de interés. La media de la precisión promedio máxima (MPPM) es el promedio de todas las PPM de los sitios de interés.

En el conjunto de imágenes de Oxford5k, las imágenes categorizadas como *Good* o *Ok* son tratadas como imágenes positivas, mientras que imágenes que no se encuentran en estos grupos son tratadas como negativas. Las imágenes categorizadas

como *Junk* son completamente ignoradas y no afectan el PPM. En el conjunto de imágenes de Google-Landmarks no tenemos una clasificación de la calidad de la imagen, por lo que simplemente evaluamos positivo si es la clasificación del sitio de interés correcto y en caso contrario es negativa.

5.3. Resultados

Se hacen experimentos con los conjuntos de datos Oxford5k y Google Landmarks, y en cada uno de ellos analizamos el MPPM con distinto número de permutaciones r, tablas hash l y vocabularios visuales. En Oxford5k evaluamos los vocabularios visuales de 150 mil, 500 mil y 1 millón de palabras visuales, $\eta = [0.02, 0.2]$[2] con incrementos de 0.02 y $r = 2$ y 3. En Google Landmarks evaluamos los 20 sitios de interés con más imágenes y hacemos varios experimentos con distintas cantidades de imágenes por sitio de interés: 500, mil, 2 mil y 3 mil. Usamos vocabularios visuales de 150 mil, 500 mil y 1 millón de palabras visuales pero solo evaluamos $\eta = 0.02$ y $r = 2$ ya que esta combinación mostró buenos resultados en Oxford5k para las distintas características. El código de todos los experimentos se puede obtener de github[3].

5.3.1. SIFT con Oxford5k y Google Landmarks

Evaluamos SMH con el vocabulario visual que provee *Oxford Visual Geometry Group*[4] para Oxford5k que tiene 1 millón de palabras visuales. Se usa $r = 3$ y

[2]En Oxford5k no se tienen resultados de η=0.02 y $r = 3$ ya que resulta muy tardado hacer SMH

[3]`https://github.com/dav006/ObjectDiscovery/`

[4]Disponible públicamente en `http://www.robots.ox.ac.uk/~vgg/data/oxbuildings/index.html`

Sitio de interés	Original	Prueba	CONF1	CONF2
All Souls	0.98	0.86	0.93	0.88
Ashmolean	0.85	0.86	0.90	0.92
Balliol	0.56	0.75	0.93	0.93
Bodleian	0.83	0.85	0.98	0.98
Christ Church	0.72	0.73	0.89	0.83
Cornmarket	0.66	0.67	0.84	0.87
Hertford	0.90	0.91	0.94	0.96
Keble	0.95	0.93	0.96	0.95
Magdalen	0.43	0.48	0.74	0.72
Pitt Rivers	1.00	1.00	1.0	1.0
Radcliffe Camera	0.98	0.93	0.96	0.94
MPPM	0.80	0.82	0.92	0.91

Cuadro 5.2: Comparación de nuestras pruebas de SMH con SIFT. Columna Original muestra los resultados de Fuentes Pineda et al. (2011), Prueba es nuestro experimento con un vocabulario visual existente, CONF1 es 1 millón de palabras visuales, $r = 2$, $\eta = 0.04$ y CONF2 1 millón, $r = 3$, $\eta = 0.14$

$\eta = 0.1$, los resultados se muestran en el cuadro 5.2 en la columna de prueba. Se puede observar que nuestro resultado es coherente con el articulo original de Fuentes Pineda et al. (2011).

Usamos nuestros propios vocabularios visuales de tamaños 150 mil, 500 mil y 1 millón de palabras visuales. Los resultados se muestran en la figura 5.3 y se puede observar que son mejores al experimento anterior y que el resultado más alto se tiene con un vocabulario de 1 millón de palabras visuales. También podemos notar que con distintos valores r los resultados son similares, los mejores resultados con CONF1 y CONF2 se encuentran en el cuadro 5.2. Entre menor sea η mejores resultados tenemos, pero llega a un máximo MPPM y a partir de eso disminuye. El problema de incrementar η es que usamos más recursos computacionales (RAM y CPU) por la necesidad de tener más tablas hash y hacer más procesamiento en las tablas.

En Google Landmarks usamos un vocabulario visual de 1 millón de palabras

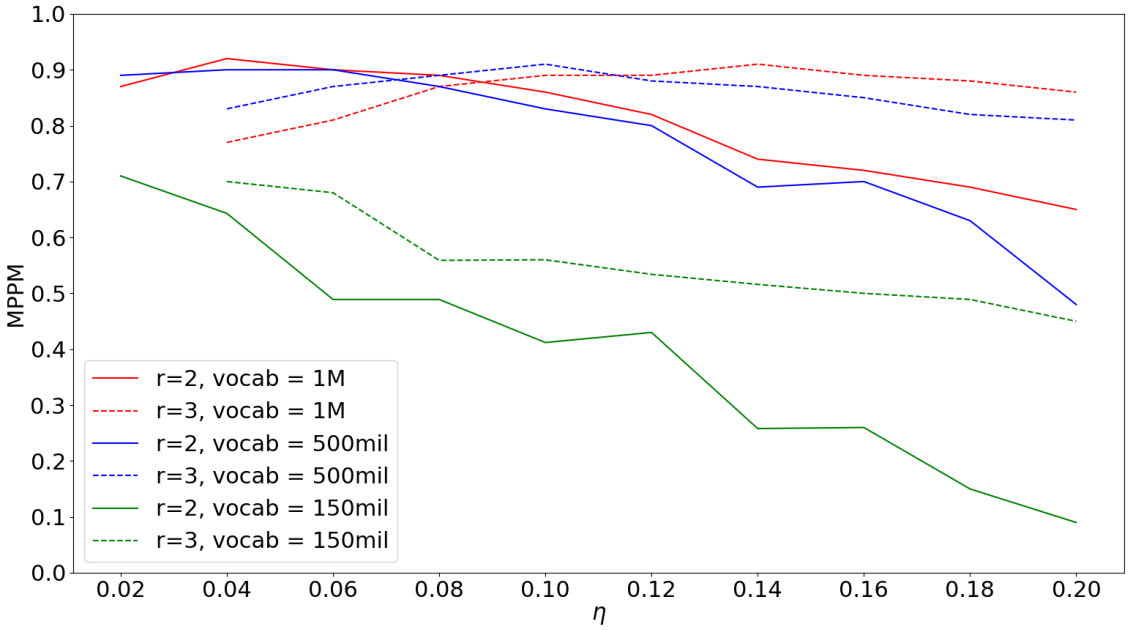

Figura 5.3: Resultados del método SMH con SIFT y Oxford5k

visuales ya que en el experimento con Oxford5k y en otros artículos (Philbin et al., 2007) este tamaño obtiene los mejores resultados. Los parámetros de SMH que usamos son $\eta = 0.02$ y $r = 2$ ya que en el experimento anterior observamos que se tienen buenos resultados con estos parámetros. Los resultados se muestran en la figura 5.4. Es interesante notar que los resultados son bajos y que al incrementar la cantidad de imágenes por sitio de interés la eficiencia del método no cambia mucho. En el anexo 6.2 se puede observar el PPM por sitio de interés y el tiempo de ejecución de SMH, que es bastante alto. Esto lo atribuimos a las siguientes razones:

1. Las imágenes de Google Landmarks tienen más resolución que las de Oxford5k por lo tanto tenemos más características. El promedio de descriptores por imagen es 2,600 y 4,300 en Oxford5k y Google Landmarks respectivamente.

2. Cada palabra del índice invertido esta contenida en promedio en 78 imágenes

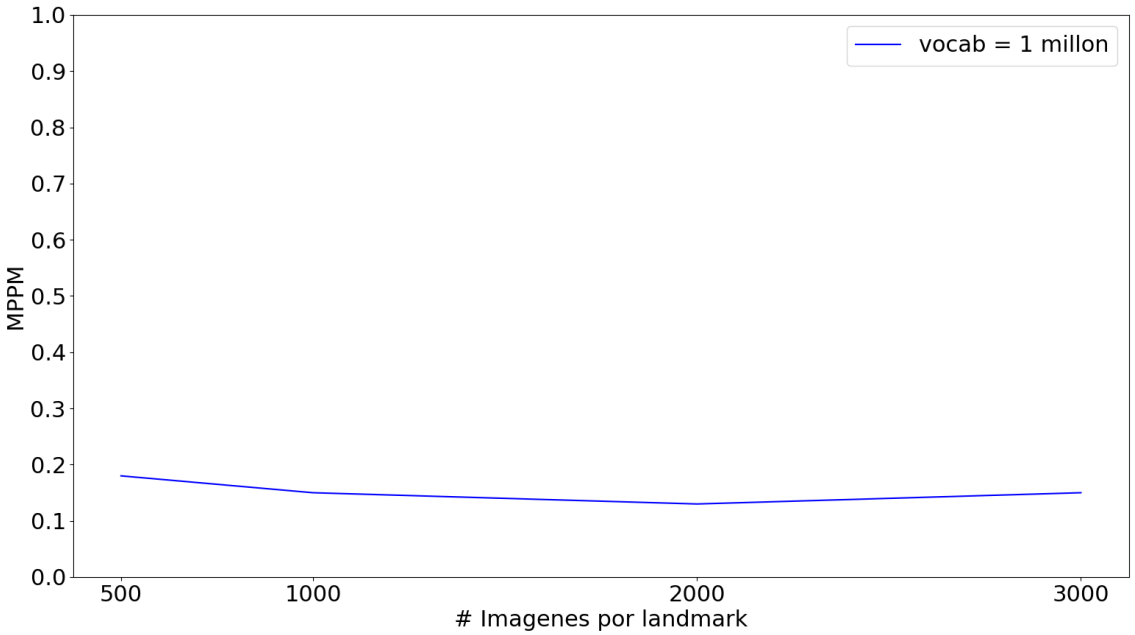

Figura 5.4: Resultados del método SMH con SIFT y Google Landmarks

por lo que es disperso. Sin embargo el método de SMH tiene problemas con descartar palabras ruidosas.

3. El vocabulario visual se construye con un muestreo de las características, debido a que hay demasiadas características para guardar en memoria. Es posible que este muestreo no este dando buenos resultados en la selección del vocabulario visual.

5.3.2. DELF con Oxford5k y Google Landmarks

Los resultados con DELF en Oxford5k se muestran en la figura 5.5. Se puede observar que el resultado más alto es con r=2 y un vocabulario de 150 mil palabras visuales. Es interesante notar que comparándolo con SIFT obtenemos mejores resultados con un vocabulario visual más pequeño. Esto creemos que se debe a que

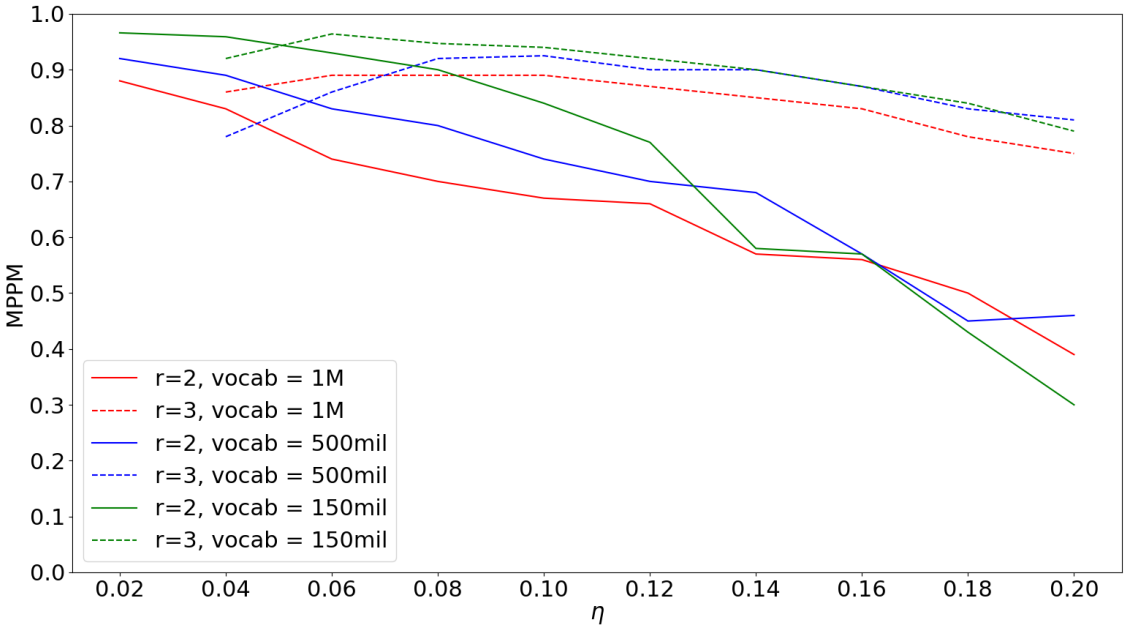

Figura 5.5: Resultados del método SMH con DELF y Oxford5k

las características son más discriminatorias por el mecanismo de atención de DELF. Similar al caso anterior, entre menor sea η mejores resultados tenemos.

En este experimento usamos menos memoria RAM y tiempo de CPU, por las siguientes razones:

1. Los descriptores DELF tienen 40 dimensiones y los descriptores SIFT tienen 128.

2. El número de descriptores DELF por imagen es menor al número de descriptores SIFT por imagen, mil y 2,600 respectivamente.

Los resultados de Google Landmarks se muestran en la figura 5.6. Es interesante notar que los resultados son mucho mejores que con SIFT y que al incrementar la cantidad de imágenes por sitio de interés el MPPM del método no cambia mucho. En el anexo 6.2 se puede observar el PPM por sitio de interés y el tiempo de ejecución

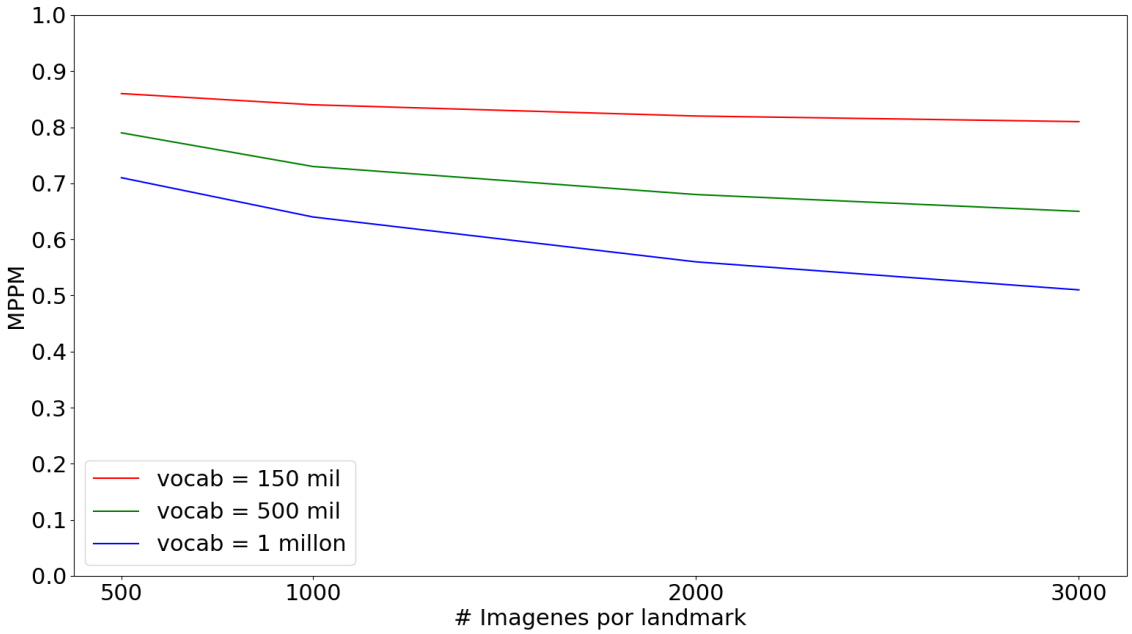

Figura 5.6: Resultados del método SMH con DELF y Google Landmarks

de SMH.

5.3.3. LIFT con Oxford5k y Google Landmarks

Los resultados de LIFT con Oxford5k se muestran en la figura 5.7. Se puede observar que los resultados son muy similares entre distintos vocabularios visuales con los mismos valores de r. El mejor resultado es MPPM de 0.8 y se obtiene con $r = 2$, $\eta = 0.02$ y un vocabulario de 150 mil palabras visuales.

Los resultados con Google Landmarks se muestran en la figura 5.8. Es interesante notar que los resultados son mejores que SIFT pero peores que DELF y que al incrementar la cantidad de datos por sitio de interés la eficiencia del método no cambia mucho. En el anexo 6.2 se puede observar el PPM por sitio de interés y el tiempo de ejecución de SMH.

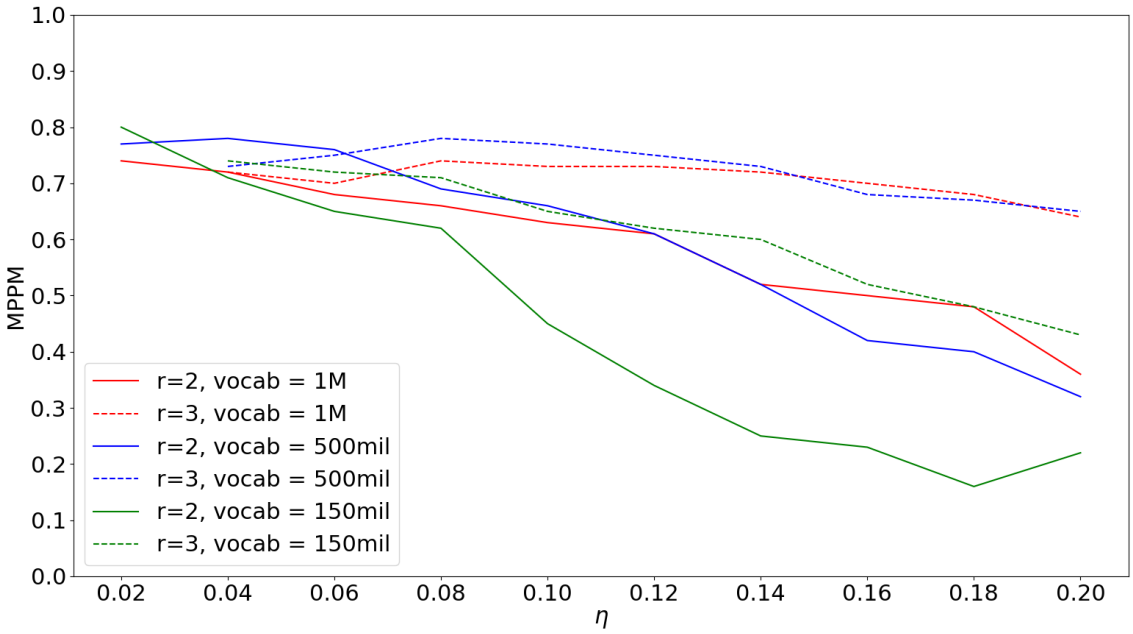

Figura 5.7: Resultados del método SMH con LIFT y Oxford5k

Aunque en Oxford5k no se vio una mejoría sobra SIFT en Google Landmarks si vemos una mejoría aunque no superior a DELF.

5.3.4. Sensibilidad de SMH a número de sitios de interés

Hacemos un análisis del comportamiento de DELF en Google Landmarks incrementando el número de sitios de interés. Usamos los 20, 40, 80 y 160 sitios de interés con más imágenes, un vocabulario visual de 150 mil, $r = 2$ y $\eta = 0.02$ por tener los mejores resultados y 500 imágenes por sitio de interés. Los resultados se pueden ver en la figura 5.9 es interesante notar que el MPPM disminuye lentamente a medida que incrementamos el número de sitios de interés, de 0.87 a 0.8. Uno de los problemas con evaluar más sitios de interés es que no tenemos los suficientes datos, ya que el conjunto de imágenes esta desbalanceado.

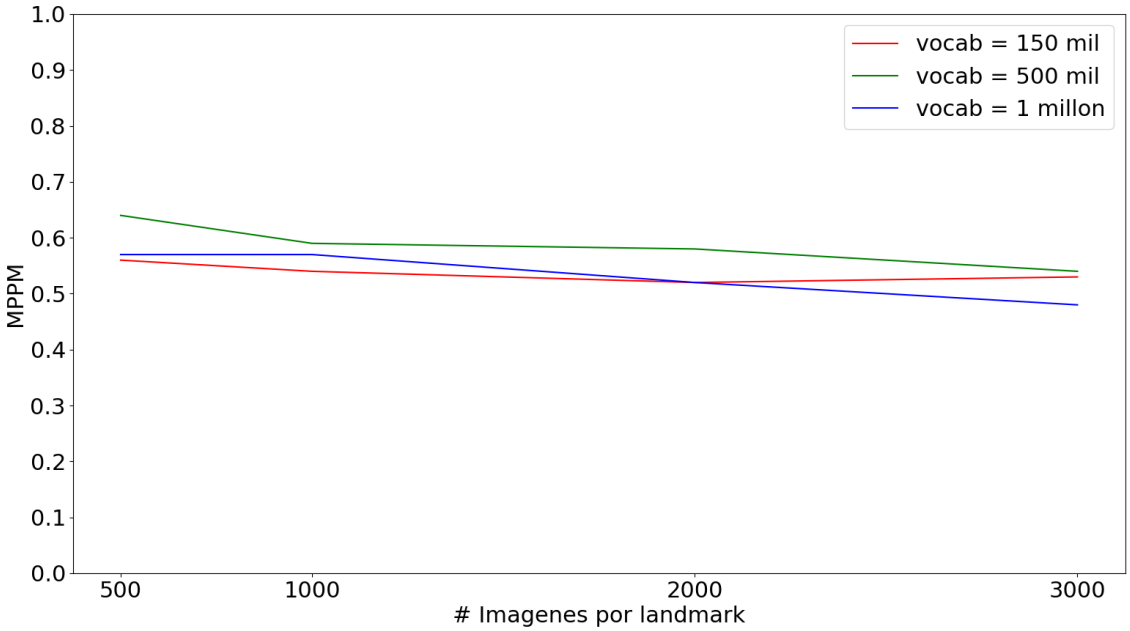

Figura 5.8: Resultados del método SMH con LIFT y Google Landmarks

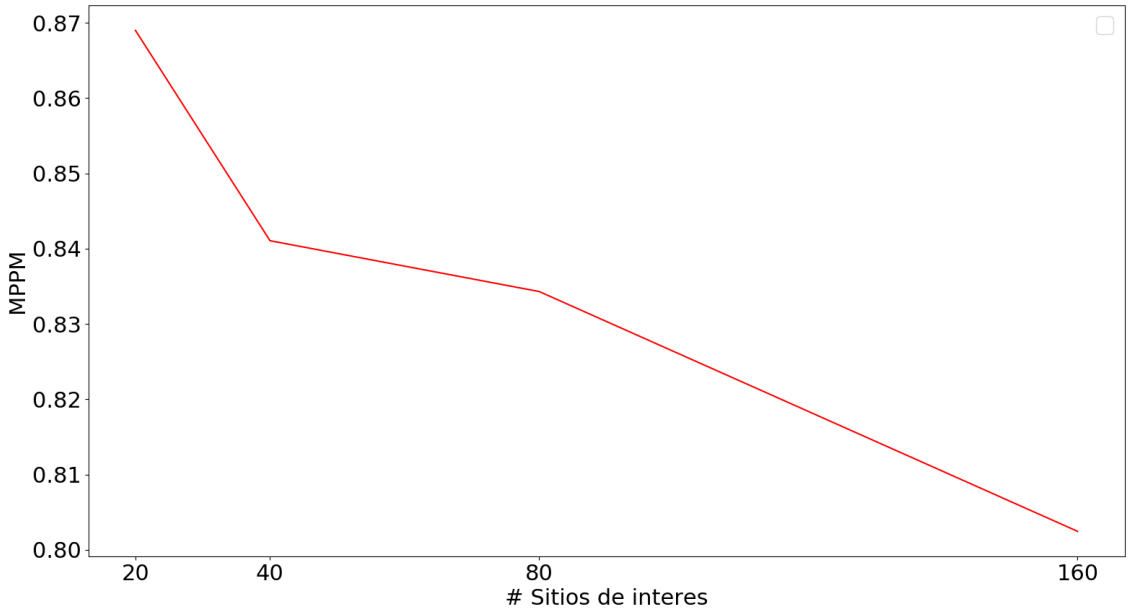

Figura 5.9: Resultados de DELF y Google Landmarks incrementando el número de sitios de interés

5.3.5. Evaluación cualitativa de Google Landmarks y DELF

A continuación mostramos resultados cualitativos de los objetos descubiertos con DELF en Google Landmarks. La figura 5.10 muestra el objeto descubierto con el máximo MPPM y algunas imágenes que son verdaderos positivos mientras que las figuras 5.11 y 5.12 muestra falsos positivos y falsos negativos respectivamente. Los puntos rojos muestran las palabras visuales que coinciden en el objeto descubierto.

Es interesante notar que el método tiene problemas para diferenciar patrones que ocurren comúnmente en ciudades. Además el conjunto de imágenes tiene la desventaja que las imágenes no fueron etiquetadas manualmente, por lo que no todas las imágenes contienen información relevante al sitio de interés, a pesar de que la foto fue tomada cerca del sitio de interés. En nuestro análisis cualitativo se muestran algunos falsos negativos que no contienen el sitio de interés, por lo que no fue un error del método.

5.3.6. Escalabilidad

El método de SMH lleva minutos para el descubrimiento de objetos de Oxford5k. Si usamos más imágenes como es el caso de Google Landmarks podemos ver buenos resultados con DELF y SIFT. El experimento que más tiempo consume en SMH fue DELF con 3 mil imágenes por sitio de interés, un vocabularios de 150 mil palabras visuales que tardó 3 horas.

El proceso de SMH es escalable ya que podemos controlar el número de tablas y permutaciones que disminuyen o aumentan el uso de recursos computacionales. El

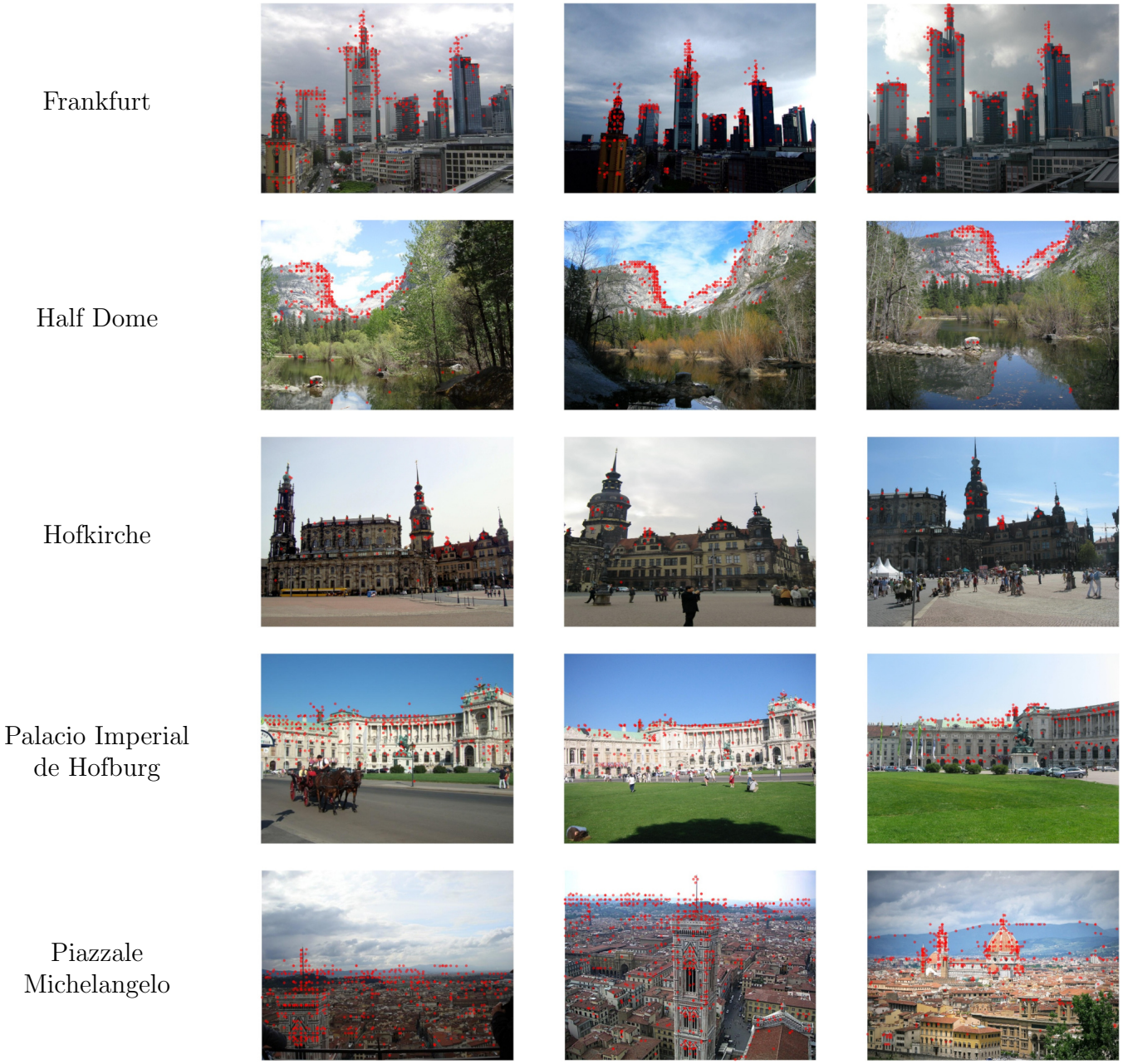

Figura 5.10: Imágenes de muestra de 5 objetos descubiertos que representan verdaderos positivos.

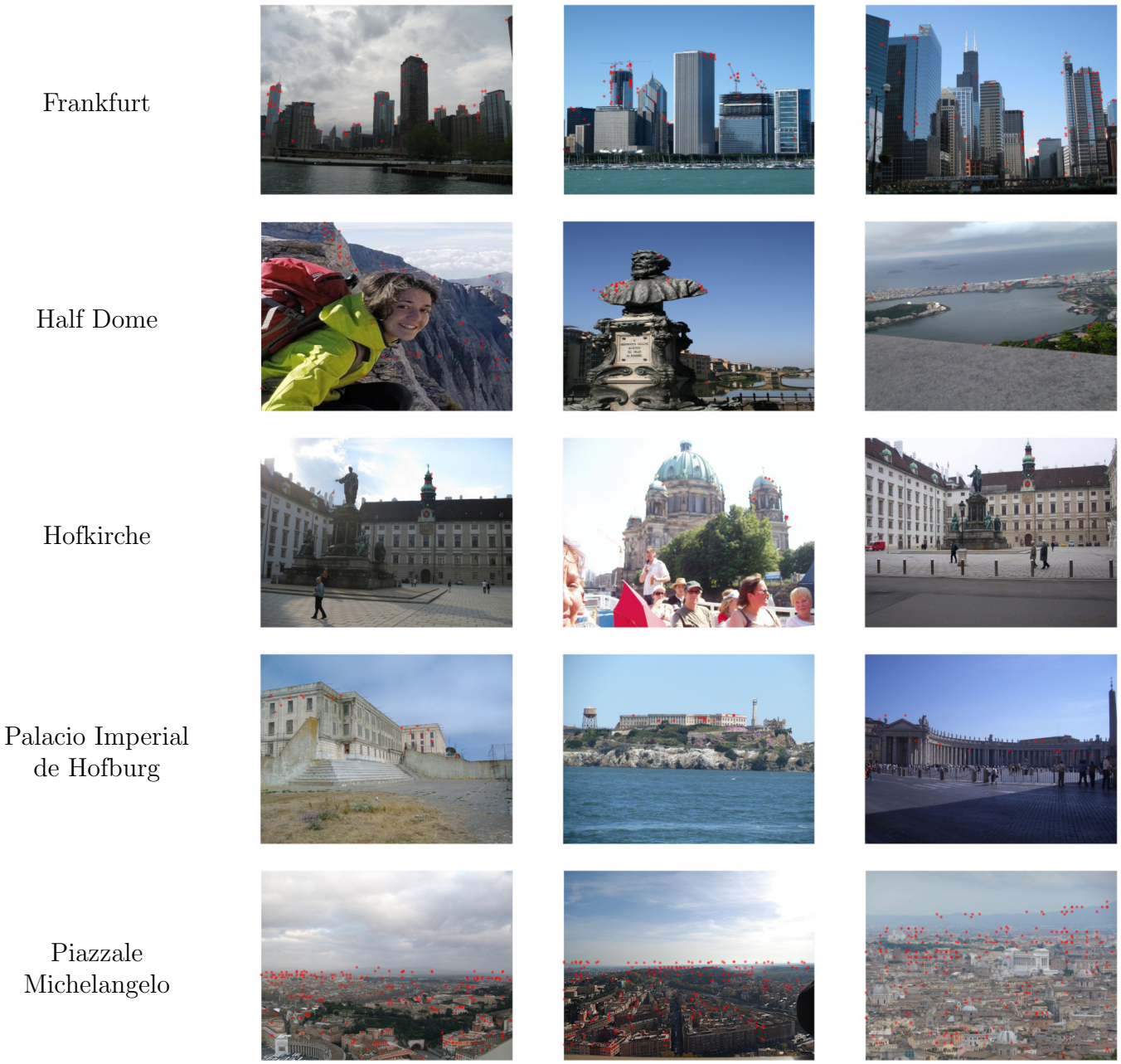

Figura 5.11: Imágenes de muestra de 5 objetos descubiertos que representan falsos positivos.

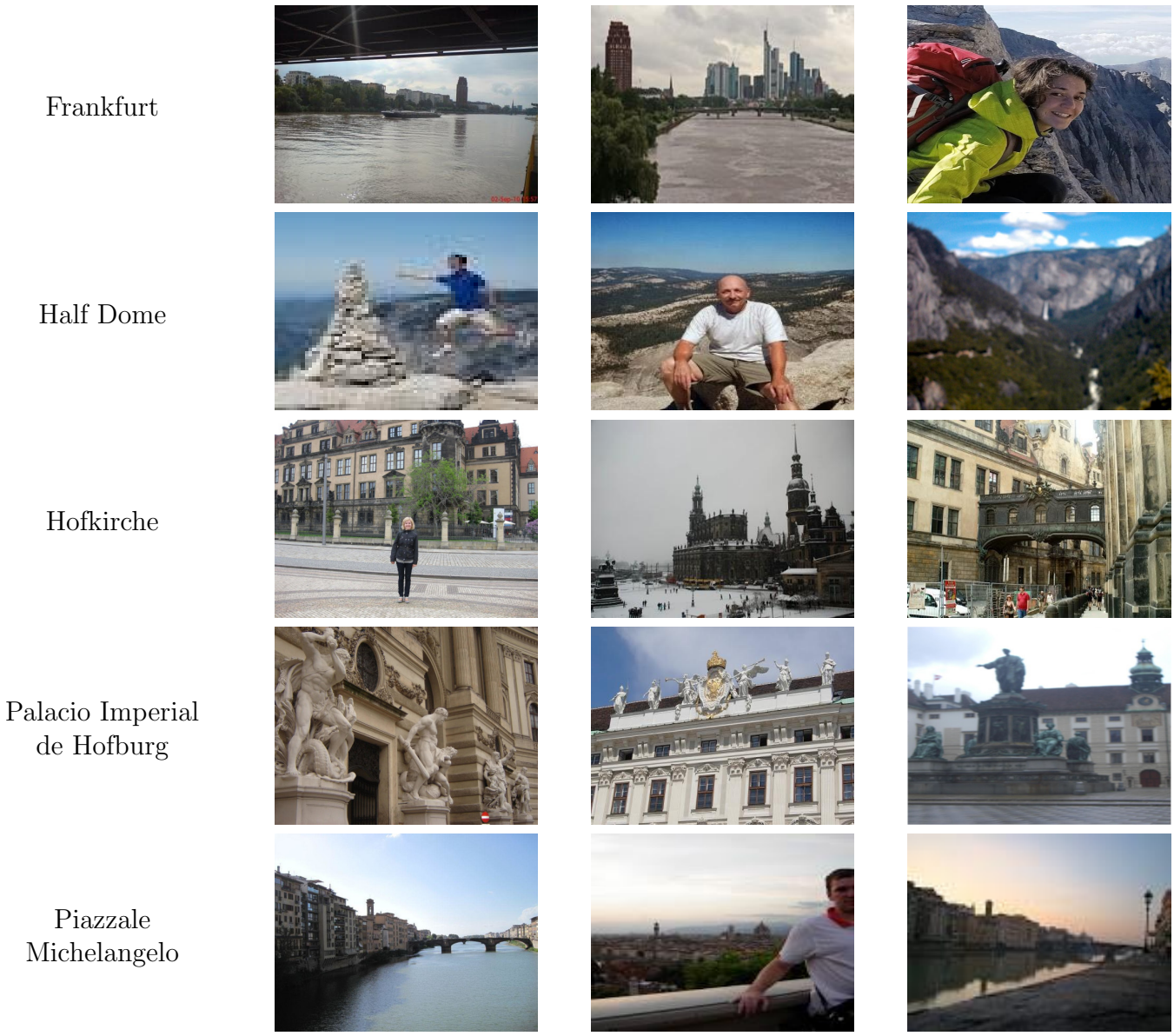

Figura 5.12: Imágenes de muestra de 5 objetos descubiertos que representan falsos negativos.

problema hacer más escalable el método es que sacrificamos la PPM en la recuperación. Identificamos problemas con Goolge Landmarks y SIFT porque el proceso puede tardar días. Uno de los problemas es la cantidad de características que se evalúan en SIFT. El proceso de descartar palabras ruidosas es lo que más tiempo consume. Obtener palabras co-ocurrentes lleva 23 min, descartar palabras ruidosas lleva 9 horas y el agrupamiento aglomerativo lleva 30 min. Esto se debe a que hay muchas colisiones por palabras co-ocurrente ruidosa, una posible opción es opción es incrementar $r = 3$ y η.

6

Conclusiones y trabajo futuro

6.1. Resumen

El método SMH muestra los mejores resultados en MPPM con características DELF para los conjuntos de imágenes Oxford5k y Google Landmarks. Además, usa menos recursos computacionales como memoria RAM y tiempo de ejecución, esto por tener menos características por imagen y de menor dimensiones. Esto se debe que estas características son más relevantes y producen palabras co-ocurrentes más estables.

En Oxford5k observamos que $r = 3$ es más estable que $r = 2$ con distintos valores de η. Es importante notar que a pesar de que $r = 3$ tiene mejores resultados con η más grandes, el número de tablas es mayor que con $r = 2$.

Las características basadas en CNN no fueron mejores que las característica manuales SIFT para el descubrimiento de instancias en todos los escenarios. Por ejemplo, LIFT no obtiene mejores resultados en Oxford5k que SIFT.

El método de SMH disminuye su MPPM muy lentamente a medida que usamos una mayor cantidad de imágenes, de 5 mil a 60 mil. El MPPM también disminuye lentamente a medida que usamos más sitios de interés. El tiempo de ejecución de SMH se mantiene en un máximo de 3 horas para los descriptores DELF y LIFT. En el caso de SIFT vimos tiempos de ejecución muy altos que requieren mayor

investigación en el proceso de descartar palabras ruidosas.

El conjunto de imágenes de Google Landmarks tiene la desventaja que las imágenes que no fueron etiquetadas manualmente, por lo que no todas las imágenes contienen información relevante al sitio de interés dando lugar a falsos negativos.

6.2. Trabajo a Futuro

Uno de los problemas de usar la representación BoW es que la creación del vocabulario visual es muy tardada, como se puede ver en los anexos, el tiempo en K-Medias puede llegar a tardarse días enteros. A pesar que usamos HNSW para la búsqueda de ANN, un trabajo a futuro sería evaluar si es posible re-utilizar vocabularios visuales existentes.

Un trabajo a futuro interesante sería incorporar más información a las características, por ejemplo, la posición de la característica o las características de sus vecinos cercanos. Esto podría ayudar a disminuir los falsos positivos en la creación de las palabras co-ocurrentes.

En la actualidad hay investigación que busca descubrir características más adecuadas para el método de agrupamiento. Un trabajo interesante es incorporar el método de SMH al proceso de extracción de características.

Se requiere hacer el proceso de descartar palabras ruidosas mas eficiente estos ocupan la mayor parte del tiempo de SMH. Quitar coocurrenta duplicados, y un analisis mas a detalle.

Bibliografía

Aumüller, M., Bernhardsson, E., y Faithfull, A. (2017). Ann-benchmarks: A benchmarking tool for approximate nearest neighbor algorithms. In *International Conference on Similarity Search and Applications*, páginas 34–49. Springer.

Babenko, A., Slesarev, A., Chigorin, A., y Lempitsky, V. (2014). Neural codes for image retrieval. In *European conference on computer vision*, páginas 584–599. Springer.

Blei, D. M., Ng, A. Y., y Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022.

Broder, A. Z. (1997). On the resemblance and containment of documents. In *Proceedings. Compression and Complexity of SEQUENCES 1997 (Cat. No. 97TB100171)*, páginas 21–29. IEEE.

Chum, O. et al. (2009a). Large-scale discovery of spatially related images. *IEEE transactions on pattern analysis and machine intelligence*, 32(2):371–377.

Chum, O., Perd'och, M., y Matas, J. (2009b). Geometric min-hashing: Finding a (thick) needle in a haystack. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, páginas 17–24. IEEE.

Cohen, E., Datar, M., Fujiwara, S., Gionis, A., Indyk, P., Motwani, R., Ullman, J. D., y Yang, C. (2001). Finding interesting associations without support pruning. *IEEE Transactions on Knowledge and Data Engineering*, 13(1):64–78.

Cook, A. (2017). Global average pooling layers for object localization. `https://alexisbcook.github.io/2017/global-average-pooling-layers-for-object-localization/`.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., y Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database.

Dumoulin, V. y Visin, F. (2016). A guide to convolution arithmetic for deep learning. *arXiv preprint arXiv:1603.07285*.

Fuentes Pineda, G., Koga, H., y Watanabe, T. (2011). Scalable object discovery: A hash-based approach to clustering co-occurring visual words. *IEICE TRANSACTIONS on Information and Systems*, 94(10):2024–2035.

Fuentes-Pineda, G. y Meza-Ruiz, I. V. (2019). Topic discovery in massive text corpora based on min-hashing. *Expert Systems with Applications*.

He, K., Zhang, X., Ren, S., y Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, páginas 770–778.

Hofmann, T. (2017). Probabilistic latent semantic indexing. In *ACM SIGIR Forum*, volumen 51, páginas 211–218. ACM.

Hubel, D. H. y Wiesel, T. N. (1959). Receptive fields of single neurones in the cat's striate cortex. *The Journal of physiology*, 148(3):574–591.

Jaderberg, M., Simonyan, K., Zisserman, A., et al. (2015). Spatial transformer networks. In *Advances in neural information processing systems*, páginas 2017–2025.

Jegou, H., Douze, M., y Schmid, C. (2008). Hamming embedding and weak geometric consistency for large scale image search. In *European conference on computer vision*, páginas 304–317. Springer.

Jégou, H., Douze, M., y Schmid, C. (2009). Packing bag-of-features. In *2009 IEEE 12th International Conference on Computer Vision*, pages 2357–2364. IEEE.

Kennedy, L. y Chang, S.-F. (2008). Internet image archaeology: Automatically tracing the manipulation history of photographs on the web. In *Proceedings of the 16th ACM international conference on Multimedia*, páginas 349–358. ACM.

Krizhevsky, A., Sutskever, I., y Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, páginas 1097–1105.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., y Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551.

Li, W., Wang, C., Zhang, L., Rui, Y., y Zhang, B. (2015). Scalable visual instance mining with instance graph. In *BMVC*, páginas 98–1.

Lindeberg, T. (1998). Feature detection with automatic scale selection. *International journal of computer vision*, 30(2):79–116.

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110.

Malkov, Y. A. y Yashunin, D. A. (2018). Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE transactions on pattern analysis and machine intelligence*.

Nister, D. y Stewenius, H. (2006). Scalable recognition with a vocabulary tree. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volumen 2, páginas 2161–2168. Ieee.

Noh, H., Araujo, A., Sim, J., Weyand, T., y Han, B. (2017). Largescale image retrieval with attentive deep local features. In *Proceedings of the IEEE International Conference on Computer Vision*, páginas 3456–3465.

Perd'och, M., Chum, O., y Matas, J. (2009). Efficient representation of local geometry for large scale object retrieval.

Philbin, J., Chum, O., Isard, M., Sivic, J., y Zisserman, A. (2007). Object retrieval with large vocabularies and fast spatial matching. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, páginas 1–8. IEEE.

Philbin, J., Sivic, J., y Zisserman, A. (2011). Geometric latent dirichlet allocation on a matching graph for large-scale image datasets. *International journal of computer vision*, 95(2):138–153.

Philbin, J. y Zisserman, A. (2008). Object mining using a matching graph on very large image collections. In *Computer Vision, Graphics & Image Processing, 2008. ICVGIP'08. Sixth Indian Conference on*, páginas 738–745. IEEE.

Quack, T., Ferrari, V., y Van Gool, L. (2006). Video mining with frequent itemset configurations. In *International Conference on Image and Video Retrieval*, páginas 360–369. Springer.

Sharif Razavian, A., Azizpour, H., Sullivan, J., y Carlsson, S. (2014). Cnn features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, páginas 806–813.

Simonyan, K. y Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Sivic, J., Russell, B. C., Efros, A. A., Zisserman, A., y Freeman, W. T. (2005). Discovering object categories in image collections.

Sivic, J. y Zisserman, A. (2003). Video google: A text retrieval approach to object matching in videos. In *null*, página 1470. IEEE.

Smeulders, A. W., Worring, M., Santini, S., Gupta, A., y Jain, R. (2000). Content-based image retrieval at the end of the early years. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (12):1349–1380.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., y Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, páginas 1–9.

Tolias, G., Sicre, R., y Jégou, H. (2015). Particular object retrieval with integral max-pooling of cnn activations. *arXiv preprint arXiv:1511.05879*.

Wang, X.-J., Xu, Z., Zhang, L., Liu, C., y Rui, Y. (2012). Towards indexing representative images on the web. In *Proceedings of the 20th ACM international conference on Multimedia*, páginas 1229–1238. ACM.

Wang, X.-J., Zhang, L., Liu, M., Li, Y., y Ma, W.-Y. (2010). Arista-image search to annotation on billions of web photos. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, páginas 2987–2994. IEEE.

Yi, K. M., Trulls, E., Lepetit, V., y Fua, P. (2016). Lift: Learned invariant feature transform. In *European Conference on Computer Vision*, páginas 467–483. Springer.

Yue-Hei Ng, J., Yang, F., y Davis, L. S. (2015). Exploiting local features from deep networks for image retrieval. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, páginas 53–61.

Zeiler, M. D. y Fergus, R. (2014). Visualizing and understanding convolutional networks. In *European conference on computer vision*, páginas 818–833. Springer.

Zhang, W., Li, H., Ngo, C.-W., y Chang, S.-F. (2014). Scalable visual instance mining with threads of features. In *Proceedings of the 22nd ACM international conference on Multimedia*, páginas 297–306. ACM.

Zheng, L., Wang, S., Wang, J., y Tian, Q. (2016a). Accurate image search with multi-scale contextual evidences. *International Journal of Computer Vision*, 120(1):1–13.

Zheng, L., Yang, Y., y Tian, Q. (2018). Sift meets cnn: A decade survey of instance retrieval. *IEEE transactions on pattern analysis and machine intelligence*, 40(5):1224–1244.

Zheng, L., Zhao, Y., Wang, S., Wang, J., y Tian, Q. (2016b). Good practice in cnn feature transfer. *arXiv preprint arXiv:1604.00133*.

Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., y Torralba, A. (2016). Learning deep features for discriminative localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, páginas 2921–2929.

Anexo A. Resultados DELF con Google Landmarks

Cuadro 6.1: PPM de Delf y 500 imágenes por sitio de interés con vocabularios visuales de 150 mil, 500 mil y 1 millón

Sitio de interés	**150 mil**	**500 mil**	**1 millón**
Frankfurt	0.690735808511	0.586571904694	0.567915386633
Half Dome	0.948658596974	0.925906068433	0.901208845744
Hofkirche	0.897426447944	0.770380816222	0.642395571932
Palacio Imperial de Hofburg	0.649730636426	0.626755715839	0.524287732883
Catedral de Berlín	0.927009138097	0.880640190887	0.766665211385
Puente Carlos	0.848810453678	0.78535450576	0.80633008056
Panteón de Agripa	0.709880635486	0.661418541803	0.629150855976
Lagoa	0.928325666393	0.820146693088	0.649806233862
Isla de Alcatraz	0.888449916766	0.74197147898	0.842136063235
Casa Batlló	0.876161186338	0.779116752093	0.613670294584
Puente de Rialto	0.960720431595	0.920500930647	0.829742072726
Torres Petronas	0.965032130566	0.946138118534	0.781996029501
Coliseo	0.916606561655	0.93825509301	0.751492470756
Ciudad de Chicago	0.762859400701	0.743689685137	0.721292578246
Piazzale Michelangelo	0.712516679751	0.686175132135	0.636933014182
MNAC	0.961249939193	0.724077361029	0.608796094075
Santa Sofia	0.934903061986	0.788610021017	0.73997519064
Palazzo Comunale	0.954038144217	0.893998930748	0.683557295144
Plaza de San Pedro	0.893514694112	0.793008380107	0.811319395577
Alhambra	0.95583993039	0.87090189198	0.758242086465
MPPM	0.86912347303895	0.794180910607	0.713345625205
Tiempo en K-Medias	3 horas	3.5 horas	4 horas
Tiempo en SMH	0.86 horas	1.14 horas	0.83 horas
# Objetos descubiertos	22893	140426	172306

Cuadro 6.2: PPM Delf y 1000 imágenes por sitio de interés con vocabularios visuales de 150 mil, 500 mil y 1 millón

Sitio de interés	**150 mil**	**500 mil**	**1 millón**
Frankfurt	0.680748067054	0.490294757153	0.427364595248
Half Dome	0.953461691252	0.929845711354	0.891143847496
Hofkirche	0.852452118952	0.734904637777	0.481526150571
Palacio Imperial de Hofburg	0.616430082221	0.595192595544	0.4941926689
Catedral de Berlín	0.916111284578	0.827498041142	0.758918539495
Puente Carlos	0.751780665192	0.658989839322	0.721578341436
Panteón de Agripa	0.719281338658	0.688831517261	0.588589738348
Lagoa	0.91925148864	0.791233470526	0.785089740917
Isla de Alcatraz	0.803639536282	0.577956871576	0.619340555643
Casa Batlló	0.845547398735	0.673470921074	0.536039880188
Puente de Rialto	0.93474655767	0.847755459766	0.687262855033
Torres Petronas	0.924509531086	0.917766839027	0.766270639815
Coliseo	0.940365139803	0.832406031202	0.711992382767
Ciudad de Chicago	0.776662849693	0.702342094925	0.694202101068
Piazzale Michelangelo	0.710261159085	0.680236183377	0.625239925972
MNAC	0.967047182778	0.796403854402	0.615398445645
Santa Sofia	0.888596610607	0.79718176406	0.633401733322
Palazzo Comunale	0.940126783277	0.831930663256	0.623079382067
Plaza de San Pedro	0.820346452	0.652254611149	0.543804296844
Alhambra	0.945696786787	0.764089999395	0.691415684767
MPPM	0.845353136217	0.739529293164	0.644792575277
Tiempo en K-Medias	6 horas	7.33 horas	8 horas
Tiempo en SMH	0.6 horas	0.77 horas	0.43 horas
# Objetos descubiertos	14,281	46,698	55,994

Cuadro 6.3: PPM de Delf y 2000 imágenes por sitio de interés con vocabularios visuales de 150 mil, 500 mil y 1 millón

Sitio de interés	**150 mil**	**500 mil**	**1 millón**
Frankfurt	0.675829367705	0.483759417335	0.347913334877
Half Dome	0.944913297648	0.84953373805	0.801443567525
Hofkirche	0.759069946739	0.579227481432	0.430222647713
Palacio Imperial de Hofburg	0.598877541457	0.561515021448	0.440746643767
Catedral de Berlín	0.908527508799	0.734627831919	0.668905156005
Puente Carlos	0.816549269753	0.552801365445	0.638445714454
Panteón de Agripa	0.70381511602	0.666862638404	0.560032564805
Lagoa	0.93157155723	0.750433721088	0.361644450593
Isla de Alcatraz	0.684615951243	0.513709555432	0.524641714827
Casa Batlló	0.862021880293	0.582459127607	0.372253574611
Puente de Rialto	0.947349721572	0.851287461599	0.759779268725
Torres Petronas	0.897491586863	0.837488306673	0.766009605412
Coliseo	0.932481742253	0.860060079976	0.633072107321
Ciudad de Chicago	0.775968238961	0.565121104002	0.634814813516
Piazzale Michelangelo	0.710787386533	0.626701675503	0.590776442573
MNAC	0.966903381037	0.672540681705	0.684956687424
Santa Sofia	0.948968417514	0.85209763761	0.768958689718
Palazzo Comunale	0.936594902519	0.845621936376	0.399854057583
Plaza de San Pedro	0.758372623313	0.698047125616	0.412492734927
Alhambra	0.822145512325	0.690831202154	0.503425204496
MPPM	0.829142747489	0.688736355469	0.565019449044
Tiempo en K-Medias	7 horas	8 horas	9 horas
Tiempo en SMH	1.18 horas	1.22 horas	0.5 horas
# Objetos descubiertos	11,355	19,658	17,802

Cuadro 6.4: PPM de Delf y 3000 imágenes por sitio de interés con vocabularios visuales de 150 mil, 500 mil y 1 millón

Sitio de interés	**150 mil**	**500 mil**	**1 millón**
Frankfurt	0.665130735211	0.525005636817	0.3297251846
Half Dome	0.92806986023	0.781490653835	0.581978679463
Hofkirche	0.792294136734	0.436483970586	0.289152183558
Palacio Imperial de Hofburg	0.580670813388	0.514845761465	0.377573392259
Catedral de Berlín	0.903126703914	0.805091135681	0.624580392654
Puente Carlos	0.744203604723	0.530267308086	0.549808815594
Panteón de Agripa	0.729089186315	0.691755775149	0.501884354173
Lagoa	0.9047608667	0.696671270575	0.33395793994
Isla de Alcatraz	0.68372851944	0.470648818475	0.548091243297
Casa Batlló	0.843300902523	0.417831547813	0.418612087753
Puente de Rialto	0.92242053589	0.811783162025	0.671425682709
Torres Petronas	0.932293360086	0.830897630862	0.682445622385
Coliseo	0.857802463879	0.599591614239	0.610285819475
Ciudad de Chicago	0.726816103251	0.587088387544	0.610710049924
Piazzale Michelangelo	0.707296726995	0.58747999695	0.505892779363
MNAC	0.96990958958	0.708820408719	0.478020426573
Santa Sofia	0.878736578605	0.82233167088	0.668422313605
Palazzo Comunale	0.927522025406	0.861235691302	0.542576159726
Plaza de San Pedro	0.768094186649	0.702866211993	0.416858093025
Alhambra	0.89430224113	0.712579156319	0.493961109392
MPPM	0.817978457032	0.654738290466	0.511798116473
Tiempo en K-Medias	9.6 horas	12 horas	15 horas
Tiempo en SMH	3 horas	1.5 horas	0.5 horas
# Objetos descubiertos	10742	14487	10778

Anexo B. Resultados LIFT con Google Landmarks

Cuadro 6.5: PPM de LIFT y 500 imágenes por sitio de interés con vocabularios visuales de 150 mil, 500 mil y 1 millón

Sitio de interés	**150 mil**	**500 mil**	**1 millón**
Frankfurt	0.283238913447	0.333656191241	0.280710842039
Half Dome	0.261131293614	0.428835688295	0.426239333765
Hofkirche	0.640658916565	0.837651009967	0.733949153132
Palacio Imperial de Hofburg	0.542391569166	0.562612157652	0.527597767489
Catedral de Berlín	0.674545247509	0.781064017994	0.684858288587
Puente Carlos	0.426605594412	0.489699602108	0.464092688475
Panteón de Agripa	0.583142361333	0.635790465777	0.628624274877
Lagoa	0.789058745372	0.909581096538	0.916780691935
Isla de Alcatraz	0.400110039325	0.512814708413	0.466963658073
Casa Batlló	0.529991852222	0.825830356365	0.607834284066
Puente de Rialto	0.832807631485	0.888555040146	0.83245349199
Torres Petronas	0.764081871723	0.623679727832	0.320483960337
Coliseo	0.217872613963	0.355273525523	0.346877512857
Ciudad de Chicago	0.318656844937	0.294616638421	0.206340843352
Piazzale Michelangelo	0.528991251106	0.595615381227	0.575442241356
MNAC	0.861300648238	0.846138145856	0.890376567945
Santa Sofia	0.578056872887	0.701988859916	0.585751773712
Palazzo Comunale	0.917251083722	0.915798088576	0.916747997289
Plaza de San Pedro	0.73800500491	0.846932826412	0.79481300012
Alhambra	0.454697459246	0.484593802499	0.381551082061
MPPM	0.567129790759	0.643536366538	0.579424472673
Tiempo en K-Medias	4.45 horas	6 horas	6.8 horas
Tiempo en SMH	0.75 horas	0.48 horas	0.33 horas
# Objetos descubiertos	8386	110823	128045

Cuadro 6.6: PPM de LIFT y 1000 imágenes por sitio de interés con vocabularios visuales de 150 mil, 500 mil y 1 millón

Sitio de interés	**150 mil**	**500 mil**	**1 millón**
Frankfurt	0.2564460266	0.345675299002	0.300793352491
Half Dome	0.287589391337	0.497405216143	0.449722064938
Hofkirche	0.620956692941	0.726925066447	0.741735776066
Palacio Imperial de Hofburg	0.536588456891	0.54436066007	0.521886026012
Catedral de Berlín	0.661650062083	0.738186345659	0.724845956188
Puente Carlos	0.458481384309	0.501671561031	0.4602447919
Panteón de Agripa	0.604838376017	0.490096505463	0.657441776367
Lagoa	0.802736528081	0.815032187362	0.908639545549
Isla de Alcatraz	0.443630631129	0.496177948516	0.539853408554
Casa Batlló	0.434746198112	0.569557526507	0.618369748025
Puente de Rialto	0.82930151743	0.848444155909	0.822638189363
Torres Petronas	0.642623777749	0.632334226005	0.27503967908
Coliseo	0.134932554011	0.319638384869	0.277810797392
Ciudad de Chicago	0.325778845989	0.317709803371	0.230252820785
Piazzale Michelangelo	0.544261548352	0.579491826491	0.566350324606
MNAC	0.832728676916	0.859015321679	0.862329601332
Santa Sofia	0.429406758932	0.590217572425	0.477123652305
Palazzo Comunale	0.879484813604	0.850020911621	0.810738865329
Plaza de San Pedro	0.731651016524	0.823513685526	0.79260534502
Alhambra	0.443747740778	0.450855823064	0.372016560049
MPPM	0.545079049889	0.599816501358	0.570521914068
Tiempo en K-Medias	5 horas	6.23 horas	6.6 horas
Tiempo en SMH	0.7 horas	0.8 horas	0.34 horas
# Objetos descubiertos	5724	40345	48401

Cuadro 6.7: PPM de LIFT y 2000 imágenes por sitio de interés con vocabularios visuales de 150 mil, 500 mil y 1 millón

Sitio de interés	**150 mil**	**500 mil**	**1 millón**
Frankfurt	0.215640761313	0.354065227626	0.270602503823
Half Dome	0.343571596319	0.505629566114	0.384128403661
Hofkirche	0.609440127922	0.786527015139	0.746396505448
Palacio Imperial de Hofburg	0.507230415207	0.524643872787	0.500680187784
Catedral de Berlín	0.568216567016	0.603662834757	0.64678751848
Puente Carlos	0.467495814707	0.475277964664	0.479818558222
Panteón de Agripa	0.616869704566	0.63991235275	0.579263008986
Lagoa	0.797762648181	0.640171100583	0.813569726558
Isla de Alcatraz	0.333597255529	0.467693694799	0.47037166793
Casa Batlló	0.494965307748	0.647690771831	0.640661021252
Puente de Rialto	0.793658374084	0.75864841947	0.773864122926
Torres Petronas	0.626234648192	0.600674498939	0.272317649632
Coliseo	0.229682145144	0.379215571444	0.274587651969
Ciudad de Chicago	0.277710743883	0.38695154501	0.144627388552
Piazzale Michelangelo	0.463930622183	0.534426273513	0.558776165847
MNAC	0.880914265681	0.83169267857	0.713333838111
Santa Sofia	0.4169992063	0.522092387618	0.448983080841
Palazzo Comunale	0.873308692484	0.828447006787	0.720038906464
Plaza de San Pedro	0.611821474322	0.758132024662	0.709199430104
Alhambra	0.454398960938	0.484892649877	0.350948145659
MPPM	0.529172466586	0.586522372847	0.524947774112
Tiempo en K-Medias	8 horas	10 horas	11.6 horas
Tiempo en SMH	0.75 horas	1 hora	0.35 horas
# Objetos descubiertos	4961	28673	18590

Cuadro 6.8: PPM de LIFT y 3000 imágenes por sitio de interés con vocabularios visuales de 150 mil, 500 mil y 1 millón

Sitio de interés	**150 mil**	**500 mil**	**1 millón**
Frankfurt	0.188631869569	0.326017463245	0.222170170216
Half Dome	0.424532873944	0.497941208828	0.373059900362
Hofkirche	0.567909425056	0.712592239497	0.690348233922
Palacio Imperial de Hofburg	0.513662049768	0.531985532896	0.498829270945
Catedral de Berlín	0.559062722979	0.506115264986	0.61256189918
Puente Carlos	0.493259984494	0.473020393447	0.480685000474
Panteón de Agripa	0.596470743569	0.601487274701	0.588649342024
Lagoa	0.840110468167	0.650420458727	0.666775308104
Isla de Alcatraz	0.399450474063	0.456277214384	0.410820026317
Casa Batlló	0.498421574474	0.622710399336	0.402513456837
Puente de Rialto	0.785681007855	0.795883625925	0.759483003546
Torres Petronas	0.657920533935	0.39184533511	0.193163611195
Coliseo	0.207702820111	0.349459105895	0.27448688847
Ciudad de Chicago	0.290577498091	0.284887246203	0.148734900715
Piazzale Michelangelo	0.501367288652	0.530449927401	0.519916824189
MNAC	0.905546973632	0.794583092496	0.678134159702
Santa Sofia	0.395395891317	0.429090045508	0.425093018625
Palazzo Comunale	0.862055217902	0.808868356666	0.725216737976
Plaza de San Pedro	0.62614405091	0.735191532478	0.699561685736
Alhambra	0.472795340092	0.426220429896	0.333363448297
MPPM	0.539334940429	0.546252307381	0.485178344342
Tiempo en K-Medias	12 horas	14.5 horas	17 horas
Tiempo en SMH	1.2 horas	1.17 horas	0.4 horas
# Objetos descubiertos	4779	19658	14109

Anexo C. Resultados SIFT con Google Landmarks

Cuadro 6.9: PPM de SIFT y 1 millón de palabras visuales con 500, mil, 2 mil y 3 mil imágenes por sitio de interés

#Imágenes por sitio de interés	**500**	**1,000**	**2,000**	**3,000**
Frankfurt	0.0892130791	0.0643985127	0.035752593	0.032404654
Half Dome	0.1291730965	0.1258239132	0.133283866	0.190268985
Hofkirche	0.0826804268	0.0428762613	0.041588991	0.033623712
Palacio Imperial de Hofburg	0.1765401193	0.1774933670	0.197685255	0.206318146
Catedral de Berlín	0.0491094380	0.0467847592	0.047364027	0.045963777
Puente Carlos	0.1407685651	0.1117930777	0.034913607	0.031052870
Panteón de Agripa	0.2620325775	0.2408142246	0.209219959	0.202958953
Lagoa	0.0616605520	0.0540367953	0.051466685	0.073597460
Isla de Alcatraz	0.1484483388	0.1285763113	0.119365388	0.116266278
Casa Batlló	0.2784321582	0.1308966466	0.162197239	0.108236873
Puente de Rialto	0.2657139397	0.2516767836	0.135391740	0.216868541
Torres Petronas	0.4566582116	0.3457343014	0.349306417	0.338868846
Coliseo	0.0697413276	0.0615374568	0.059597323	0.059018042
Ciudad de Chicago	0.2938265600	0.2656485539	0.184523619	0.197408719
Piazzale Michelangelo	0.0949474542	0.0804217566	0.060266013	0.046884101
MNAC	0.1722027631	0.0859081524	0.035683929	0.091012526
Santa Sofia	0.0595754823	0.0481265407	0.044484642	0.047003241
Palazzo Comunale	0.5668428803	0.4908621546	0.506789496	0.621469696
Plaza de San Pedro	0.1282671020	0.0937111886	0.039594650	0.156891866
Alhambra	0.1893327642	0.2007761730	0.218840876	0.198276684
MPPM	0.1857583418	0.1523948465	0.133365816	0.150719699
Tiempo en K-Medias	13.7 horas	16 horas	16 horas	8 horas
Tiempo en SMH	6.11 horas	10 horas	1 dia	2.2 dias
# Objetos descubiertos	4304	699	784	764