UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

INVESTIGACIÓN DE MÉTODOS EFICACES PARA BÚSQUEDAS POR SIMILITUD EN BASES DE DATOS MULTIMEDIA

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL EN COMPUTACIÓN

JAIME VÉLIZ GUTIÉRREZ

PROFESOR GUÍA:
BENJAMIN BUSTOS CÁRDENAS

MIEMBROS DE LA COMISIÓN:
CLAUDIO GUTIÉRREZ GALLARDO
EDUARDO GODOY VEGA

SANTIAGO DE CHILE
MARZO 2009

ESTE TRABAJO HA SIDO FINANCIADO POR EL PROYECTO FONDECYT 11070037.

Índice general

Índice de figuras

Índice de cuadros

Resumen

El presente trabajo está enfocado en las bases de datos multimedia, área importante de investigación en los últimos años. Por objetos multimedia se entiende contenido como audio, imágenes, objetos 3D, etc. A diferencia de las bases de datos tradicionales, en una base de datos multimedia no se buscan los objetos que cumplan una condición exacta, sino los más parecidos a un objeto conocido de antemano, llamado *objeto de consulta*. Por esta razón, la búsqueda por similitud es uno de los temas principales de estudio en las bases de datos multimedia.

Para reflejar la similitud entre objetos multimedia, en general existen distintos métodos para calcular una distancia entre dos objetos multimedia, los cuales intentan reflejar el parecido entre dichos objetos (a mayor distancia, más disímiles son). Se ha comprobado empíricamente que algunos métodos funcionan mejor que otros ante distintos tipos de objetos de consulta, y que combinar varios de ellos a la vez entrega generalmente mejores resultados que utilizar cada uno en forma separada. Sin embargo, aún no existe una forma clara de decidir la forma de combinarlos en el momento de la consulta, y muchas veces se termina utilizando el mejor método en promedio, lo cual no siempre es la mejor opción.

En este trabajo se estudia un método formal para combinar estos métodos, realizando una consulta previa en un conjunto de objetos conocido como *conjunto de entrenamiento*, que permite estimar en forma dinámica la calidad de cada método ante un objeto de consulta dado. Como primera tarea, se implementó un entorno de trabajo que permite realizar experimentos de manera masiva, incorporar fácilmente distintos tipos de bases de datos y definir los tipos de experimentos a ejecutar. El entorno de trabajo implementado permitió estudiar cómo afecta la selección de un conjunto de entrenamiento en determinar la calidad de un método de transformación de objetos multimedia.

De los resultados experimentales obtenidos, se concluye que un conjunto de entrenamiento adecuado es de un tamaño bastante pequeño en relación a la base de datos original, lo que permite estimar la calidad de un método de cálculo de similitud con poco esfuerzo adicional. Adicionalmente, se observó que el cálculo dinámico de esta información permite aproximarse bastante a mejores combinaciones de métodos, las cuales sólo pueden ser obtenidas mediante fuerza bruta. Se muestra también que algunos resultados tienden a ser independientes de la base de datos, mientras que otros son exclusivos de la forma en que ésta haya sido construida, y que en general todos los resultados poseen un buen grado de robustez, lo que permite reproducirlos fácilmente en bases de datos con distintos tipos de información multimedia.

Capítulo 1

Introducción

1.1. Reseña histórica y Contexto

El área de las bases de datos multimedia (o MMDBs) ha experimentado un gran crecimiento en los últimos años, donde la generación, procesamiento y almacenamiento de contenido multimedia y el envío de este contenido a través de redes se ha expandido rápidamente a áreas de aplicación diversas como entretenimiento, industria, ingeniería, medicina y ciencias sociales, entre otros. La gran cantidad y tamaño de objetos multimedia procesados por aplicaciones en estas áreas sugieren el uso de algún tipo de base de datos, ya que con esto se garantiza la consistencia, concurrencia, integridad, seguridad y disponibilidad de grandes volúmenes de datos, así como la existencia de métodos para consultas y recuperaciones eficientes de contenido de interés para el usuario. Las MMDBs requieren las características principales de las bases de datos tradicionales, más algunas propiedades nuevas, como métodos unificados para almacenar y procesar una gran cantidad de datos heterogéneos, así como protocolos para transmitir dicha información a las aplicaciones clientes que la requieran.

Las MMDBs incorporan características nuevas en comparación a las bases de datos tradicionales. Una de las más importantes es tener que tratar con objetos heterogéneos y generalmente complejos, donde las búsquedas en forma exacta pierden sentido. El enfoque de las MMDBs es distinto, debido a razones prácticas:

- Al hacer una búsqueda exacta, la probabilidad de que dos objetos multimedia sean exactamente iguales es casi nula (a menos que sean copias digitales), por lo tanto, los métodos existentes en las bases de datos tradicionales no pueden ser aplicados de igual forma en este contexto.

- De la misma manera, las consultas más comunes en una MMDB son las consultas por similitud, en las que el usuario tiene un objeto multimedia de referencia, y desea encontrar otros objetos lo más parecidos posible a dicho objeto.

Por esta razón, en las MMDBs se hace necesario definir formalmente esta noción de similitud, además de mantener una implementación eficiente de ella.

1.2. Motivación: Eficiencia versus Efectividad

En el área de las bases de datos multimedia, dos aspectos fundamentales que son considerados para analizar el desempeño de la búsqueda y el resultado obtenido son la eficiencia y efectividad. El primero de ellos tiene que ver con la rapidez con que la respuesta es entregada al usuario. Este aspecto es bien conocido desde las bases de datos tradicionales, donde las técnicas como indexamiento y optimización del tiempo de acceso a disco siempre se consideran en la implementación de bases de datos eficientes. El segundo tema, la efectividad, tiene que ver con la similitud entre los objetos retornados y el objeto de consulta. Este aspecto no existe en las bases de datos tradicionales, ya que ellas están diseñadas para el manejo y recuperación de *datos*. Es decir, para una consulta dada, simplemente se entregan todos los registros que cumplan con lo que indica esta consulta. El paradigma de las MMDBs es otro: Se desea entregar al usuario la información que, si bien no sea exacta, sea lo más representativa de lo que él quiere recuperar. Esto se considera recuperación de *información*, en lugar de recuperación de datos.

¿Por qué estudiar la efectividad? La eficiencia, en general, se dedica a recuperar en forma rápida objetos, empleando algoritmos y estructuras de datos apropiadas. La efectividad no cuenta con ese privilegio, ya que debe tener en cuenta el carácter inexacto y muchas veces subjetivo de la consulta, y qué tan bien satisface la respuesta al usuario. Obviamente éste quiere que la respuesta sea rápida y que lo obtenido le sea de utilidad. Si el usuario sólo recibe respuestas rápidas, pero poco coherentes, todo el sistema es deficiente. Más aún, la efectividad no se puede mejorar utilizando más recursos, como en el caso de la eficiencia, donde un aumento en ésta se puede lograr utilizando hardware proporcionalmente más rápido. De aquí surge el interés en buscar formas de mejorar la efectividad de las consultas multimedia, por ser un aspecto aún en sus etapas iniciales, y con un respaldo histórico y técnico todavía escaso.

1.3. Consultas en un MMDB mediante vectores característicos

Para realizar búsquedas en una MMDB, el método más utilizado en la actualidad es transformar cada objeto a un vector de dimensión alta, llamado vector característico, o FV (del término en inglés *"Feature Vector"*). Cada transformación existente corresponde a un algoritmo que intenta rescatar los atributos más importantes de un objeto. De esta manera, una búsqueda por similitud se implementa buscando los vectores más cercanos a un vector fijo.

En la actualidad, el método anterior presenta un mejor rendimiento si se utiliza más de un vector a la vez para calcular la distancia entre dos objetos. Se puede calcular una distancia general como una combinación lineal de las distancias individuales, donde los pesos asignados a cada uno indican su calidad como vector. Estos pesos son difíciles de calcular en forma eficaz, ya que la calidad de un vector depende del tipo de objeto de consulta (un mismo vector puede funcionar mejor o peor ante objetos distintos).

Para resolver este problema, una solución es utilizar conjuntos de entrenamiento, que consisten en grupos relativamente pequeños de objetos, en los cuales se realiza una consulta previa para estimar la calidad de un vector ante un determinado objeto, y por lo tanto, asignarle un peso adecuado. Al ser un conjunto pequeño, esto puede ser calculado rápidamente sin afectar el rendimiento de la consulta original.

1.4. Objetivos del estudio

Como objetivos generales, esta memoria pretende determinar hasta qué punto un conjunto de entrenamiento nos puede ayudar a estimar a priori qué tan buena es una medida de similitud, y si existe alguna forma de relacionar el tamaño de este conjunto con el de la MMDB original, por ejemplo un porcentaje de la cantidad original de datos, o un cierto valor máximo, a partir del cual el resultado no varía mucho. Junto con esto, se pretende aclarar un aspecto todavía no estudiado a fondo en esta área, que es cómo pueden influir los conjuntos de entrenamiento en la efectividad de una búsqueda por similitud.

Como objetivos específicos, podemos mencionar los siguientes:

1. Crear un entorno de pruebas que cumpla con los requisitos necesarios para llegar a conclusiones válidas y comparables con otros resultados dentro de la comunidad científica.
2. Ser capaces de reproducir en nuestro entorno de pruebas otros resultados conocidos de antemano, para definir un buen punto de partida al realizar nuestras propias mediciones.
3. Permitir la incorporación de bases de datos de distintos tipos de objetos multimedia (al menos dos), para evaluar si los resultados obtenidos dependen o no del tipo de objeto multimedia.

1.5. Desafíos a abordar

Aunque parezca que el conjunto de entrenamiento es una variable cualquiera que uno se haya animado a estudiar, en realidad no lo es, debido principalmente a las siguientes razones:

- En primer lugar, la ejecución masiva de experimentos con distintos conjuntos de entrenamiento, el posterior análisis numérico de los resultados y su uso para inferir cuáles son los buenos tamaños de conjunto de entrenamiento es sumamente complejo, y para que estos resultados sean válidos y comparables con otros estudios similares en la comunidad científica, se requiere la construcción de un entorno de trabajo que sea lo más parecido posible a los empleados por otros investigadores en esta área. Además, las bases de datos para hacer los experimentos correspondientes son muy escasas, lo que dificulta la posibilidad de deducir resultados válidos para cualquier tipo de objeto multimedia.

- Por un lado más teórico, el estudio particular sobre conjuntos de entrenamiento y prueba para analizar el resultado sobre la efectividad es un ámbito bastante nuevo y aún no se han realizado experimentos para decidir si existen relaciones entre estas dos variables, y en caso de existir, qué factores determinan esta relación. Dado que esta es un área todavía muy nueva y abierta a nuevos descubrimientos, la posibilidad de encontrar variables y relaciones que puedan mejorar la efectividad de una consulta es una gran motivación para realizar investigaciones sobre el tema.

El resto de este trabajo se dividirá de la siguiente manera: El Capítulo 2 presentará los conceptos necesarios para comprender el contexto de las bases de datos multimedia y el esquema experimental que será utilizado. El Capítulo 3 explicará la forma general en que los experimentos serán ejecutados y evaluados, además de indicar las herramientas ya existentes y las que se deben incorporar para este propósito. El Capítulo 4 detallará la ejecución de experimentos para la obtención de resultados ya conocidos y verificar el buen funcionamiento del entorno de pruebas. El Capítulo 5 presentará los experimentos realizados para la búsqueda de propiedades y relaciones nuevas utilizando conjuntos de entrenamiento, y el análisis general de todos los resultados obtenidos. Finalmente, en el Capítulo 6 se presentarán los resultados globales deducidos de todo el trabajo anterior y se propondrán extensiones o mejoras posibles para realizar a futuro.

Capítulo 2

Conceptos básicos

Para una mejor comprensión del informe de aquí en adelante, es necesario indicar los conceptos que el lector debe tener presente en todo momento. Estos conceptos serán utilizados frecuentemente durante este trabajo.

2.1. Vectores característicos

Como fue mencionado en la introducción, un objeto multimedia es transformado a un vector de dimensión alta para poder ser comparado con otros objetos de manera eficiente. Este vector pretende recuperar distintos atributos importantes del objeto, y corresponde a una representación aproximada del objeto dentro de un espacio vectorial de dimensión d ($\mathbb{R}^d$). Acompañando este espacio con una métrica o función de distancia δ, podemos calcular la distancia entre dos FV's, como una forma de aproximar la noción intuitiva de similitud o cercanía entre dos objetos multimedia (mientras más cercanos sean los objetos, más parecidos son, lo que se refleja en una distancia pequeña entre sus respectivos FV's). Si x_1 y x_2 son dos objetos multimedia pertenecientes a un mismo universo de objetos Ω (por ejemplo, Ω puede ser el conjunto de todas las imágenes válidas), y $t : \Omega \to \mathbb{R}^d$ es una función de transformación de estos objetos a un FV, la distancia D entre estos objetos se puede escribir formalmente como:

$$D(x_1, x_2) = \delta(t(x_1), t(x_2))$$

La distancia que ocuparemos en este trabajo será la métrica L_1, o *métrica de Manhattan* entre dos vectores $x, y \in \mathbb{R}^d$:

$$\delta_1(x, y) = \sum_{i=1}^{d} |x_i - y_i|$$

Para poder comparar y/o combinar FV's de diferente dimensionalidad, esta distancia la normalizaremos por la dimensión del respectivo FV.

2.2. Clases de objetos

Con la noción de distancia ya definida, podemos definir clases de objetos sobre una MMDB. Las clases corresponden a agrupaciones de objetos que resultan similares entre sí, y se intenta, idealmente, que las distancias entre sus respectivos FV's sean pequeñas comparadas con las distancias a objetos de otras clases. En MMDBs reales, los objetos tienden a formar grupos o clases más o menos bien definidas (por ejemplo, imágenes únicamente de animales o de objetos hechos por el hombre). En este caso, los objetos pueden ser clasificados ya sea manualmente por expertos que decidan la clase de cierto objeto, o automáticamente, cuando a cada objeto se le asigna aquella clase cuyos objetos están a la menor distancia posible del mismo. Con este concepto se pueden definir los siguientes conjuntos de objetos:

1. **Conjunto de entrenamiento (o *base de datos de entrenamiento*)**: Formalizando este concepto mencionado en la introducción, un conjunto de entrenamiento corresponde a un conjunto en el cual los objetos ya se encuentran clasificados. Por ejemplo, para una base de datos compuesta de miles de imágenes de automóviles y de aeroplanos, un conjunto de entrenamiento podría tener unas pocas decenas de imágenes de cada tipo (las que mejor representen a sus respectivas clases). Este conjunto, previamente clasificado, es lo que comúnmente se conoce como *ground truth*, es decir, es un conocimiento base que tenemos sobre qué objetos pertenecen a qué clases. Esta base de conocimiento generalmente se determina en forma manual por medio de expertos.

2. **Conjunto de prueba**: Corresponde a un conjunto disjunto del conjunto de entrenamiento, el cual también se encuentra previamente clasificado. Este conjunto posee diversas finalidades, aunque generalmente se utiliza para comprobar de manera externa los resultados obtenidos en un conjunto de entrenamiento (es posible que resultados obtenidos

con un conjunto de datos no sean válidos si utilizamos otro conjunto). En este trabajo se utilizará como el conjunto de objetos de consulta, cuyo objetivo es determinar la calidad de una búsqueda por similitud cuando se realiza sobre una MMDB real.

2.3. Objetos relevantes y métricas de evaluación

Al hacer una búsqueda por similitud, idealmente se deben obtener todos los objetos de la misma clase que el objeto de consulta. Éste es el concepto de *objetos relevantes* para una consulta. En una MMDB, este conjunto contiene todos los objetos de la misma clase que el objeto de consulta. Por ejemplo, si una MMDB posee 10 modelos de cuerpos humanos y 15 modelos de piezas de automóviles, una consulta por figuras humanas posee 10 objetos relevantes, que son los 10 modelos que deberían ser retornados idealmente.

Para hacer una evaluación numérica de los resultados entregados por una búsqueda por similitud, tomando como referencia el caso ideal en que se obtienen todos los objetos relevantes, existen métodos convencionales usados en la actualidad. En este trabajo se utilizarán los siguientes:

1. *Diagrama Precision v/s Recall* [4]: Este indicador entrega una noción sobre la cantidad de objetos relevantes que estamos recuperando en una consulta. Supongamos que una consulta posee un conjunto R de objetos relevantes, y la búsqueda nos entrega un conjunto A como respuesta. Sea Ra la intersección de estos dos conjuntos (es decir, los objetos relevantes obtenidos en la consulta). Se definen los siguientes valores:

   - *Precision*: Es la fracción de los objetos retornados (el conjunto A) que son relevantes:

   $$Precision = \frac{|Ra|}{|A|}$$

   - *Recall*: Es la fracción de los objetos relevantes (el conjunto R) que fueron retornados en la consulta:

   $$Recall = \frac{|Ra|}{|R|}$$

   El aumento de uno de estos indicadores en general implica una disminución en el otro. Por ejemplo una consulta que siempre retorna todos los objetos de la base de datos posee

un valor de *Recall* igual a 1 (obtenemos siempre todos los objetos relevantes), pero un valor muy pequeño de *Precision* (una proporción muy baja de los objetos retornados son relevantes). Lo que se persigue generalmente es mantener estos dos indicadores en un valor alto, es decir, una consulta debe entregar todos los objetos relevantes que pueda, evitando traer demasiados objetos no relevantes. Para eso, este diagrama muestra los valores de *Precision* a medida que aumenta el valor de *Recall*. Supongamos que una consulta posee 10 objetos relevantes, y una búsqueda por similitud nos entrega 15 objetos, de los cuales aquellos en las posiciones 1, 4, 6, 11 y 15 son relevantes. Entonces el diagrama *Precision v/s Recall* indica que hay 100 % de *Precision* al 10 % de *Recall* (cuando hemos recuperado el primer 10 % de objetos relevantes, el 100 % del total de objetos obtenidos hasta ese momento es relevante). Cuando obtenemos el segundo objeto relevante (el cuarto en la lista), decimos que tenemos 50 % de *Precision* al 20 % de *Recall* (cuando hemos visto 2 de los 10 objetos relevantes, el 50 % de los objetos vistos hasta ese momento son relevantes).

Este diagrama permite visualizar en un sólo gráfico el comportamiento de ambos indicadores, *Precision* y *Recall*, y está asociado a una consulta en particular, es decir, para otra consulta por similitud este diagrama será distinto. Un ejemplo de este diagrama se ilustra en la Figura 2.1, donde a medida que aumenta *Recall*, el valor de *Precision* disminuye.

2. *R-Precision* [4]: Este indicador está bastante relacionado con el anterior, ya que pretende comparar los valores de *Precision* y *Recall* en un sólo valor. Si una consulta posee R objetos relevantes, el valor de *R-Precision* indica el valor de *Precision* cuando se han visto los primeros R objetos retornados por la consulta. Volviendo al ejemplo del punto anterior, como existen 10 objetos relevantes, corresponde revisar las primeras 10 posiciones en la respuesta. De estas posiciones, los objetos en las posiciones 1, 4 y 6 son relevantes, por lo tanto, el valor de *R-Precision* en este caso sería de 30 %. Un alto valor de este indicador implica que la consulta entrega rápidamente la mayoría de los objetos relevantes, con pocos objetos no relevantes dentro de ellos.

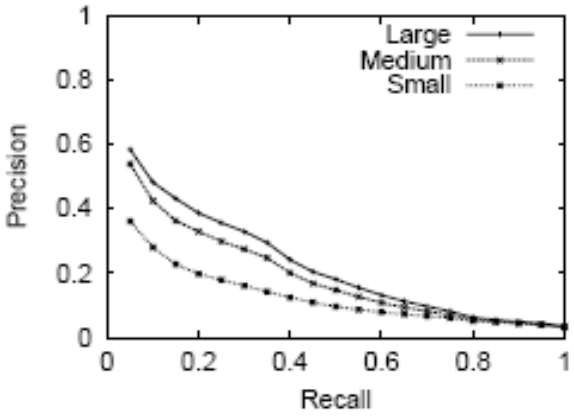

Figura 2.1: Ejemplo de diagrama Precision v/s Recall. A medida que recuperamos objetos relevantes en la consulta, generalmente se recuperan objetos no relevantes junto con ellos, lo que hace disminuir el valor de *Precision*.

2.3.1. Combinación de funciones de similitud

En la mayoría de los tipos de objetos multimedia actuales, existen distintas representaciones como FV, cada una de ellas dedicada a representar de la forma más precisa posible la similitud entre objetos. Aunque cada FV posee un desempeño diferente a los demás, en la práctica se observa que combinar varios FV al mismo tiempo puede entregar mejores respuestas que utilizando cada uno en forma individual [2]. Para dos objetos multimedia distintos O_1 y O_2, supongamos que existen k representaciones en forma de FV. Si $d_1, \ldots, d_k$ denotan las respectivas distancias entre los objetos usando cada FV (eventualmente normalizadas según la dimensionalidad del mismo), entonces es posible calcular una distancia global d combinando las distancias individuales con un determinado peso para cada una:

$$d = \sum_{i=1}^{k} w_i d_i \ ,$$

donde los pesos w_i deben ser escogidos de manera adecuada. En este trabajo, los pesos w_i serán calculados evaluando la calidad de la respuesta entregada por el FV, utilizando un conjunto de entrenamiento que nos permita estimar cómo sería la consulta en la MMDB real.

Para evaluar la calidad de un determinado FV, se utilizará el indicador *Entropy Impurity* [3].

Este indicador determina la coherencia de una respuesta según qué clases de objetos contiene ésta. Si todos los objetos que obtenemos son de la misma clase, este indicador es igual a 0 (no hay impureza en la respuesta), mientras que si todos los objetos pertenecen a distintas clases, este indicador alcanza su valor máximo. Formalmente, si $P = \{p_1, p_2, \ldots, p_n\}$ es un conjunto de probabilidades, tal que $\sum_{i=1}^{n} p_i = 1$, el índice de *entropy impurity* $I(P)$ se calcula como:

$$I(P) = -\sum_{i=1}^{n} p_i \log(p_i)$$

donde en el caso nuestro, las frecuencias son las fracciones de objetos en la respuesta que corresponden a cada clase (sólo se consideran las clases que tienen al menos una aparición en la respuesta). Este indicador se encuentra basado en el conjunto de entrenamiento escogido, ya que una búsqueda por similitud nos entrega objetos de dicho conjunto. La clasificación de estos objetos nos indica qué tan bien recuperamos los objetos de la clase correcta, es decir, qué tan bien fue escogido el conjunto de entrenamiento.

Al calcular el indicador $I(P)$ en el conjunto de entrenamiento, el peso que utilizaremos para el respectivo FV será el siguiente:

$$w = \log(m) - I(P)$$

donde m es el número de objetos retornados en la consulta. Cuando no hay impureza en la respuesta, este peso alcanza su valor máximo, igual a $\log(m)$, y si la impureza es máxima, este peso es igual a 0.

2.4. Relación con el trabajo a realizar

En este trabajo, la finalidad de la base de datos de entrenamiento es determinar la calidad de la medida de similitud que utilicemos para calcular distancias entre objetos. Es decir, teniendo este conjunto definido, una buena medida de similitud debería entregar vectores cercanos entre sí para objetos de la misma clase, y distantes para objetos de clases distintas. Al determinar la calidad de una medida de similitud, será posible escoger los pesos adecuados para combinar estas métricas y aumentar la efectividad de las consultas en una MMDB.

Para definir el conjunto de entrenamiento adecuado para nuestros propósitos, se utilizará el índice de *entropy impurity* recién descrito, ya que éste nos permitirá decidir si las respuestas

obtenidas en una consulta se acercan al ideal esperado (la mayoría de los objetos pertenecientes a la clase del objeto de consulta). Si este comportamiento se repite en una gran cantidad de objetos de consulta, entonces se ha encontrado un buen conjunto de entrenamiento para determinar la calidad de una medida de similitud.

2.5. Estado del arte en el área

Para realizar nuestro estudio, tendremos como base algunos resultados fundamentales que se han comprobado en el área de las MMDBs. Por ejemplo, para cada tipo de dato multimedia (imagen, texto, objetos 3D, etc.) existen métodos para transformar los objetos a representaciones vectoriales, que funcionan bastante bien en la práctica, como se muestra en [1]. También se ha comprobado que el uso de más de un tipo de FV, combinados en alguna manera inteligente, es más efectivo que utilizando el mejor FV individual al momento de decidir la clase de un objeto, indicado en [2]. Esto no se limita sólo a combinaciones de vectores en un espacio vectorial. En efecto, la combinación de vectores es parte de un método más general, en el que también es posible combinar métricas de distintos espacios métricos. En estos espacios, un objeto multimedia no necesariamente se representa como un vector en $\mathbb{R}^d$, sino que puede ser un cierto valor en otro espacio métrico.

Otro resultado comprobado empíricamente [1], es que no existe un mejor FV para todos los objetos en general. Como éstos capturan distintos atributos de los objetos multimedia, un FV puede ser mejor o peor según el tipo de consulta y el tipo de objeto. Por ejemplo, un FV que se dedica principalmente a capturar la distribución de color de una imagen puede ser muy malo si el usuario en realidad está interesado en los contornos y formas de una imagen o si la imagen es en blanco y negro, pero en otros casos puede ser un FV muy bueno.

Finalmente, un resultado empírico comprobado en la actualidad es que la mejora en efectividad generalmente presenta un punto de saturación. Por ejemplo, como se comprueba en [1], la efectividad mejora mientras más dimensiones (más atributos) rescatamos de un objeto, pero llegando a un cierto valor, el agregar más dimensiones mejora muy poco, o nada, este resultado.

Con este marco de referencia, queremos ver si la relación entre la base de datos de entrenamiento elegida y la efectividad lograda a partir de ella depende de variables como el tamaño

de la base de datos de entrenamiento, el número de clases contenidas en ella o el número de objetos por clase, y si algunas de estas variables son más importantes en ciertas situaciones que en otras, como en el caso de los FV's, los cuales podían variar su efectividad en función del tipo de objeto y del tipo de consulta.

2.6. Caso particular: Búsqueda de objetos 3D

2.6.1. Características principales

Para ejemplificar el problema de la búsqueda en bases de datos multimedia, se detallará un caso particular: La búsqueda de objetos 3D. Este problema se ha extendido rápidamente en áreas como medicina o entretenimiento. Según el dominio de la aplicación, los objetos 3D pueden ser almacenados en una variedad de formatos, de los cuales uno de los más comunes es representarlos como una malla de polígonos definidos por un conjunto de vértices. Esta información puramente geométrica es la que generalmente se tiene al momento de realizar una búsqueda por similitud. A diferencia de otros tipos de datos multimedia, los objetos 3D pueden ser relativamente complejos tanto en las estructuras de datos utilizadas para su representación, como en los métodos empleados para visualizarlos gráficamente. Esto hace más costosa la búsqueda de objetos similares entre sí, ya que los FV's existentes para este tipo de objetos deben tener en cuenta dificultades adicionales que solucionar. Dentro de estas dificultades, los FV's necesitan incorporar las siguientes características deseables para búsqueda de objetos 3D:

1. *Invariancia ante operaciones geométricas*: Dos objetos 3D, aunque sean parecidos según el punto de vista del usuario, pueden ser almacenados con distinta orientación, tamaño, desplazamiento con respecto a un cierto punto de origen, etc. Una propiedad que debería tener un buen FV para tratar estos casos es la invariancia con respecto a la posición en que sea almacenado un objeto. De esta manera, una consulta tendrá un buen desempeño sin importar la posición de los objetos en la MMDB. Estas propiedades en general son logradas aplicando transformaciones geométricas sobre los objetos. Algunos métodos utilizados con frecuencia son definir el origen en el centro de masa del objeto, definir sus ejes principales utilizando algún método estadístico conocido, y normalizando las coordenadas de sus vértices para eliminar el factor tamaño.

2. *Invariancia ante nivel de detalle*: Un objeto 3D puede tener múltiples niveles de detalle, según lo requiera una cierta aplicación. Si un mismo objeto posee representaciones distintas, con distintos grados de detalle, un FV bien construido debe ser robusto ante este caso. Es decir, reconocer objetos similares aunque éstos se encuentren definidos con una distinta cantidad de vértices o polígonos, por ejemplo.

2.6.2. La base de datos *Princeton Shape Benchmark*

Dado que los FV's asociados a objetos 3D son complejos en comparación a otro tipo de datos multimedia, resulta útil probar el desempeño de ellos en primera instancia. Para comenzar nuestro estudio, la primera MMDB que utilizaremos será de objetos 3D. Una de las más difundidas para investigación en el área es la base de datos conocida como *Princeton Shape Benchmark* (o *PSB*) [5]. Esta base de datos consiste en alrededor de 1.814 objetos (modelos 3D) previamente clasificados, los cuales se dividen en 907 objetos de test y 907 objetos de prueba, intentando mantener la misma cantidad de clases y objetos en cada conjunto. Los objetos en esta base de datos pertenecen a clases encontradas fácilmente en el mundo real, como por ejemplo animales, muebles o figuras humanas. No se incluyen objetos de dominios específicos, como piezas de maquinaria o biología molecular. Por último, esta base de datos posee distintos niveles de clasificación, formando una jerarquía en que la clasificación más gruesa separa los objetos en dos clases; objetos naturales y hechos por el hombre, y la clasificación más fina contiene objetos agrupados según propiedades geométricas mucho más específicas. Para efectos de nuestro estudio, utilizaremos esta última clasificación, ya que es la que mejor permite evaluar la efectividad de una consulta al retornar objetos de clases específicas.

2.6.3. Descriptores para objetos 3D

En la actualidad, los descriptores propuestos para representar objetos 3D intentan resolver los problemas descritos anteriormente utilizando distintas estrategias, todas ellas aplicables a representaciones de objetos 3D como una malla de polígonos. Sin embargo, todos aplican el mismo proceso para pasar de la representación como objeto 3D a un vector de dimensión alta:

1. *Normalización de las coordenadas del objeto*. Para proveer las propiedades deseadas de invariancia ante transformaciones del objeto tales como rotación o escala, las coordenadas del objeto son transformadas de la manera que sea necesaria. Por ejemplo, se pueden utilizar técnicas como PCA (*Principal Component Analysis*) para encontrar los ejes principales del objeto, desplazar el objeto para que su centro de masa quede ubicado

en el origen del sistema de coordenadas, y normalizar sus coordenadas para que el objeto quede totalmente contenido dentro de un cubo de tamaño fijo.

2. *Representación abstracta del objeto*. Una vez normalizado el objeto, su representación puede ser interpretada de distintas formas. Las formas más comunes de abstraer un objeto 3D (representado generalmente como una malla de polígonos) son como una superficie que delimita el volumen de un objeto en 3D, como una superficie infinitamente delgada con propiedades geométricas bien definidas (por ejemplo, una superficie cerrada), o como proyecciones sobre un conjunto de planos, produciendo mapas de contorno o profundidad del objeto para representar su superficie.

3. *Transformación numérica*. Dependiendo del tipo de abstracción seleccionada, el paso a información numérica puede ser realizada de distintas formas. Por ejemplo, las particiones del espacio o las imágenes proyectadas sobre un plano pueden ser transformadas utilizando métodos como transformaciones discretas de Fourier, mientras que las representaciones como superficie se pueden transformar calculando medidas sobre puntos escogidos sobre la superficie del objeto. La información numérica obtenida de esta forma en general implica perder algo de información sobre el objeto.

4. *Generación del descriptor*. Para representar la información numérica en forma de un vector de dimensión alta, existen muchos enfoques, dependiendo del algoritmo utilizado. Para los vectores utilizados en este trabajo existen dos formas principales: Una de ellas es crear directamente el FV a partir de valores que representen atributos importantes del objeto, tales como curvatura de la superficie o volumen total ocupado. La otra estrategia es crear un histograma tomando las distribuciones de ciertos atributos específicos, por ejemplo, distancias entre puntos aleatorios ubicados en la superficie del objeto, y posteriormente transformar el histograma a una representación vectorial.

El proceso general de creación de un FV se ilustra en la Figura 2.2. En base a la estrategia utilizada, los descriptores pueden ser clasificados en 3 grupos distintos, los cuales se explican con detalle en [1], junto a la descripción más detallada del proceso de creación de un descriptor.

Descriptores basados en Volumen

Este grupo de descriptores pretende recuperar la información global sobre el objeto mediante el cálculo de muestras sobre regiones más pequeñas. Para eso, el esquema general es

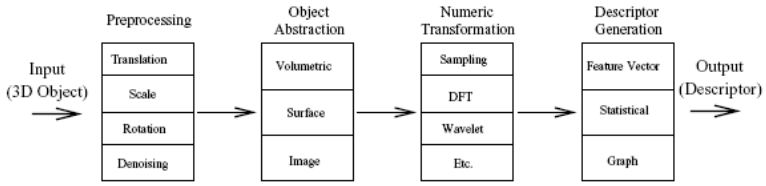

Figura 2.2: Proceso general de creación de un FV, a partir de un objeto representado generalmente como una malla de polígonos [1].

particionar el espacio (previamente normalizado para proveer invariancia ante rotación o escala del objeto) en varias subregiones, y sobre cada una de ellas calcular alguna medida apropiada, por ejemplo la cantidad de volumen o superficie del objeto contenido en dicha región. El FV resultante generalmente contiene una dimensión por cada subregión utilizada. La forma más común de particionar el espacio es utilizar una malla homogénea de cubos (o *voxels*), aunque existen otras formas más sofisticadas. Por ejemplo, un descriptor basado en el volumen ocupado por un objeto 3D utiliza el cubo que contiene al objeto (conocido como *bounding cube*), dividiendo cada una de las 6 superficies en n^2 cuadrados cada una, y generando $6n^2$ pirámides uniendo estas superficies con el centro del cubo. La Figura 2.3 ilustra esta técnica en 2 dimensiones.

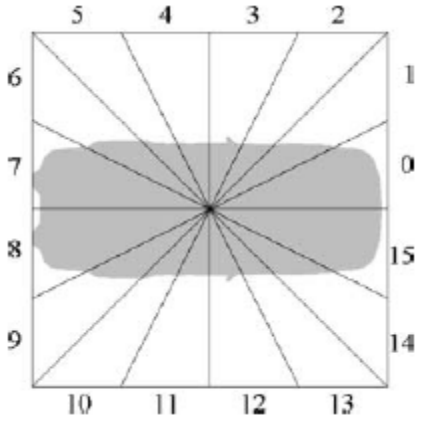

Figura 2.3: Esquema de partición del espacio para uno de los descriptores basados en volumen.

Descriptores basados en Superficie

Este grupo se basa principalmente en obtener información sobre el objeto mediante cálculos sobre elementos ubicados en la superficie del objeto, como vértices, caras, puntos ubicados aleatoriamente sobre la superficie, etc. Teniendo únicamente esta información, los descriptores intentan obtener información sobre la forma del objeto, generalmente analizando la curvatura de su superficie. Por ejemplo, el FV denominado *Cords-based* define un *cord* como un vector que va desde el centro de masa del objeto hasta un punto representativo de la superficie (generalmente el centroide de un triángulo que forma parte de la malla de polígonos). El descriptor se construye en base a la distribución de las longitudes de los *cords*, así como las distribuciones de los ángulos que éstos forman con los ejes principales del objeto. Otro descriptor conocido como *Shape distribution with D2*, o SD2, utiliza distribuciones de probabilidad para reflejar propiedades geométricas del objeto. Algunas de las distribuciones utilizadas son, por ejemplo, los ángulos entre 3 puntos aleatorios de la superficie, las distancias entre 2 puntos aleatorios en la superficie, o el área del triángulo formado por 3 puntos aleatorios en la superficie. Finalmente, conviene mencionar al descriptor *Shape spectrum*, el cual en lugar de calcular estadísticas a nivel agregado sobre el objeto, construye el vector característico en base a las curvaturas locales calculadas sobre puntos distribuidos en la superficie del objeto. Como la curvatura es calculada localmente, este vector es robusto ante objetos articulados, como por ejemplo, figuras humanas con las extremidades en distintas posiciones.

Descriptores basados en Imágenes

Este tipo de descriptores obtiene información sobre un objeto proyectándolo generalmente sobre los planos paralelos a los ejes principales. La información sobre el objeto es obtenida desde su superficie, en forma de imágenes que son transformadas posteriormente por algún método estándar, generando los valores numéricos para el FV respectivo. Por ejemplo, el FV denominado *Silhouette* determina los contornos del objeto proyectado en cada plano principal, generando 3 imágenes distintas, las cuales son llevadas a una representación en un espacio distinto mediante la transformación discreta de *Fourier* en 2 dimensiones. Algunos de los coeficientes de bajas frecuencias de esta transformación son los que crean el descriptor final. Otro descriptor, conocido como *Depth buffer*, sigue la misma idea, pero en lugar de generar imágenes describiendo el contorno del objeto, se crean imágenes en escala de grises, describiendo la profundidad del objeto visto desde cada plano principal. Estas imágenes siguen un proceso similar al del descriptor *Silhouette* para ser transformadas a un vector característico. Las Figuras 2.4 y 2.5 ilustran el proceso empleado por cada uno de estos métodos.

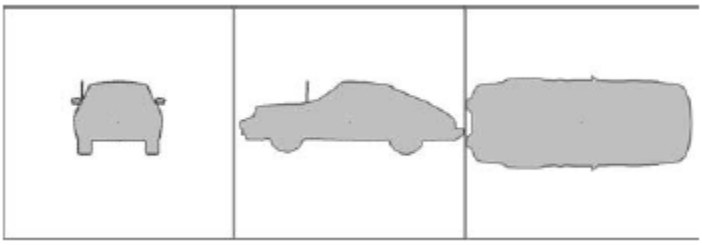

Figura 2.4: Contornos de un automóvil para el descriptor *Silhouette*. En cada plano principal se genera una imagen describiendo el contorno del objeto, la cual es transformada posteriormente para obtener valores numéricos que generan el FV respectivo.

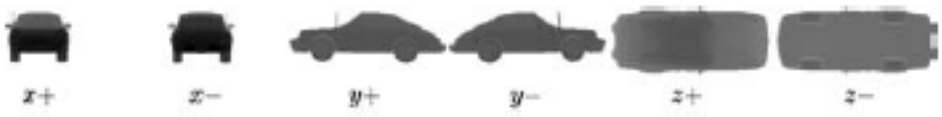

Figura 2.5: Contornos en escala de grises para el descriptor *Depth buffer*. Cada imagen almacena información sobre la profundidad del objeto visto desde esa perspectiva. La transformación a valores numéricos es similar a la que realiza el descriptor *Silhouette*.

Capítulo 3

Metodología de trabajo

3.1. Entorno de pruebas

Para efectuar los experimentos necesarios en este análisis y estudio, se construirá un entorno de pruebas. Esto significa, recopilar bases de datos multimedia adecuadas para hacer los experimentos necesarios, en lo posible recomendadas por la comunidad científica, y que se encuentren previamente clasificadas. Para hacer los experimentos con estas bases de datos ocuparemos su representación vectorial, ya sea obteniendo los objetos de la base de datos y su transformación a FV's al mismo tiempo, o utilizando herramientas disponibles para hacer la transformación directamente, sin tener que hacerlo nosotros. La representación original de los objetos multimedia la utilizaremos para tener una mejor visión gráfica de los objetos que retorna cada consulta.

El entorno de pruebas tendrá el siguiente esquema general de trabajo:

1. Inicialmente, seleccionamos un conjunto de entrenamiento adecuado. La forma de elegirlo se hará en forma sistemática, variando aspectos como el número de clases que la componen, el número de objetos por clase, etc.

2. Sobre esta base de datos de entrenamiento, tomaremos distintos objetos de la base de datos de prueba para ser utilizados como objetos de consulta. Entonces recuperaremos los n objetos más cercanos a dicho objeto en el conjunto de entrenamiento, según la función de distancia entre FV's. El parámetro n corresponde a la cantidad de objetos que serán recuperados para evaluar la coherencia de la respuesta entregada por cada FV.

3. Para el conjunto obtenido en el punto anterior, obtenemos la clase a la que pertenece cada objeto, es decir, la clasificación hecha por *ground truth* para cada uno de ellos. Con

esto obtendremos la frecuencia de aparición de cada clase dentro de nuestra respuesta. Cabe destacar que en la práctica no todos los objetos recuperados serán de la misma clase.

4. Teniendo la frecuencia de aparición de cada clase en la respuesta, calculamos el peso del FV correspondiente, utilizado el método *entropy impurity*. Repitiendo este paso para todos los FV existentes, obtendremos los pesos para cada uno de ellos, lo cual permitirá combinarlos para realizar la consulta real en la MMDB correspondiente.

5. Finalmente, sobre esta última consulta aplicaremos las métricas de eficacia (*R-Precision* y *Precision v/s Recall*) descritas en la Sección 2.3, y obtendremos la información numérica y visual, si ésta es posible, para determinar si existen las relaciones que pretendemos encontrar.

3.2. Herramientas existentes

Para lograr nuestro objetivo existen algunas herramientas para facilitar el análisis de los datos obtenidos en nuestros experimentos. Comenzaremos ocupando la base de datos *PSB*, descrita anteriormente. En el sitio web correspondiente a esta base de datos [5], se proporciona, además de los objetos mismos, una serie de utilidades de código libre, de las cuales utilizaremos las siguientes:

1. *Visualizador de objetos 3D*: Para tener una visión gráfica de los objetos de cada clase, y de los que obtenemos durante las consultas, esta utilidad permite visualizar un modelo 3D definido en un archivo de formato .off (*object file format*, formato estándar para representar objetos 3D, descrito en [7]). Con esto podemos visualizar los objetos que ocupamos de consulta o los que obtenemos en una búsqueda por similitud.

2. *Representación vectorial de objetos 3D*: La PSB no provee la representación vectorial de los objetos a los FV's más comunes utilizados en la actualidad. éstos se encuentran disponibles en otro sitio en la web [6], y definen su propio formato, el cual indica la dimensión de los FV's y una lista de todos los objetos, indicando a qué objeto de la PSB y a qué clase corresponde, junto con el valor de su respectivo FV. Actualmente existen 11 FV's diferentes calculados en archivos distintos para los objetos de la PSB.

3.3. Herramientas a incorporar

Con las herramientas recién descritas, es posible reutilizar algunas de ellas para trabajar con diversas bases de datos, no sólo la de objetos 3D. Los aspectos comunes a todas las bases de datos serán mantenidos de la siguiente forma:

1. *Datos de entrada*: Las bases de datos de objetos, junto con sus clasificaciones y FV's respectivos, pueden venir en distintos formatos. Para permitir la incorporación de varios formatos distintos, definiremos un formato abstracto en nuestro entorno de pruebas, y en cada base de datos sólo necesitaremos implementar la transformación a dicho formato. A grandes rasgos, un objeto multimedia es representado como un objeto que posee un conjunto de vectores con valores de punto flotante, y un valor entero indicando a qué clase pertenece.

2. *Separación de conjuntos de prueba y entrenamiento*: Dentro de nuestro entorno de pruebas es necesario incorporar la selección del conjunto de entrenamiento adecuado. Para esto, teniendo cada objeto previamente clasificado, crearemos distintos filtros para ellos, lo que permitirá seleccionar, entre otras posibilidades, las clases que posean un número mínimo de objetos, o mantener todas las clases con un número equilibrado de objetos. Estos filtros serán la base para determinar si existen las relaciones que pretendemos encontrar entre FV's y conjuntos de entrenamiento.

3. *Búsquedas por similitud*: Teniendo un formato abstracto de clasificación de objetos y sus FV's, la búsqueda por similitud se implementará tomando un objeto de consulta dado, calculando su distancia a todos los objetos de la base de datos de entrenamiento escogida, y manteniendo un ranking de estos valores para cada objeto consultado. De esta manera, podemos comparar los n objetos relevantes de la base de datos con los primeros n objetos del ranking, para hacer los cálculos correspondientes.

4. *Gráficos Precision v/s Recall*: En cada experimento realizado, en general obtendremos un cierto gráfico representativo del resultado obtenido. Para poder almacenar estos resultados en una forma independiente de la herramienta utilizada para generar los gráficos,

utilizaremos un formato simple y estándar para los archivos que serán procesados posteriormente. En este caso utilizaremos un formato simple como valores numéricos separados por algún delimitador, el cual puede ser procesado fácilmente en alguna aplicación externa, como por ejemplo, *Gnuplot*.

5. *Normalización de las coordenadas de los descriptores*: Cada algoritmo de creación de un FV puede entregar valores numéricos de distintos órdenes de magnitud. Por ejemplo, mientras un FV podría determinar atributos con valores entre 0 y 1, otro podría calcular estadísticas a nivel agregado con valores numéricos más grandes. Esto no es problema al evaluar cada FV por separado, sin embargo la evaluación de las combinaciones de ellos se puede ver afectada por esta diferencia (por ejemplo, al combinar todas las distancias con pesos igual a 1, predominarían las distancias más grandes). Para evitar este problema, la base de datos debe ser normalizada con alguna estrategia bien definida. La que será utilizada en este trabajo será dividir las coordenadas de un FV por un mismo valor, de manera que todos los objetos posean sus FV con coordenadas entre 0 y 1. Esto se logra preprocesando cada FV y buscando la coordenada más grande existente entre todos los objetos. Además, las distancias deben ser normalizadas por la dimensionalidad del vector, ya que la distancia L_1 aumenta con vectores de dimensión alta. Cabe mencionar que otros tipos de normalización podrían ser utilizados para mejorar la evaluación de cada FV en igualdad de condiciones.

Dado que el entorno de pruebas se encuentra bien definido para la PSB, nuestro primer objetivo directo antes de comenzar la investigación propiamente tal, es comprobar que el entorno de pruebas logra los mismos resultados conocidos en la actualidad, es decir, verificar que algunos FV's funcionan mejor que otros ante distintos tipos de objetos, y que la combinación de ellos en general entrega mejores resultados que la evaluación de un FV individual. Esto nos permitirá tener una base sólida de partida para el resto de las pruebas a realizar.

3.4. Trabajo adicional

Para poder extender nuestro trabajo a otros tipos de bases de datos, resulta indispensable mantener una implementación del entorno de pruebas que permita procesar objetos de distinta naturaleza. Aunque en principio trabajaremos sobre objetos 3D, se espera incorporar otro

tipo de objetos a nuestro entorno de pruebas. Actualmente se dispone de una base de datos de aproximadamente 150.000 imágenes, que consisten en archivos *xml* con información sobre la ubicación de la imagen y 5 FV's diferentes. El procesamiento de estos datos se implementará después de trabajar con la base de datos de objetos 3D, pudiendo ser fácilmente agregados al esquema recién descrito.

Capítulo 4

Experimentos exploratorios

En esta primera parte de nuestro trabajo se puso especial énfasis en ejecutar diversas pruebas para obtener resultados locales, los cuales son comparados con resultados obtenidos por otros investigadores en condiciones similares. Cabe destacar que los resultados no necesariamente deben ser iguales, dado que hay factores que alteran levemente los resultados entre un experimento y otro (algunos ejemplos son la dimensionalidad de los FV elegidos, la forma de combinar estos mismos, o la función de distancia utilizada). Sin embargo, el obtener resultados coherentes y cercanos a los conocidos, es suficiente para verificar que el entorno de pruebas funciona de forma correcta.

4.1. Desempeño de vectores individuales

Dentro de los primeros experimentos, uno de ellos consiste en buscar aquellos FV que poseen un mejor desempeño general que los demás. La forma de obtener esta información consiste en realizar sobre una misma base de datos una serie de búsquedas por similitud, utilizando cada objeto de la base de datos, y luego hacer el cálculo del *R-Precision* para cada consulta. El *R-Precision* global para el FV se calcula como el promedio sobre todas las consultas hechas. Finalmente, tendremos un ranking global de desempeño de cada FV, el cual podremos comparar con algun otro ranking conocido.

En las condiciones de nuestro ambiente de prueba, la Tabla 4.1 muestra el orden de desempeño de los FV's que estamos utilizando. Este ranking fue calculado realizando las consultas en el conjunto de entrenamiento de la *PSB*. Al mismo tiempo, en la Figura 4.1 se muestra el diagrama *Precision v/s Recall* para el mejor FV individual.

Comparando este ranking con el de la Tabla 4.2 obtenido en [1], se observa un buen grado de semejanza tanto en el orden en que aparecen los vectores de acuerdo a su desempeño, como en el valor de *R-Precision* obtenido para cada uno de ellos. Cabe mencionar que no todos los vectores que aparecen allí estaban disponibles para nuestros experimentos y que dichos valores se obtuvieron con otra MMDB (aunque similar a la PSB).

Además de este análisis global de cada FV, también se ha hecho un análisis local por cada clase, confirmando otro hecho conocido: Un mismo FV puede ser mejor o peor según el tipo de objeto de consulta. También se observa que existen clases de objetos "difíciles" de obtener, es decir, todos los FV entregan sistemáticamente un bajo valor de *R-Precision*. Las Tablas 4.3, 4.4 y 4.5 muestran claros ejemplos de este comportamiento.

4.2. Combinaciones de 2 ó más FV's

Un resultado conocido empíricamente, indica que utilizar más de un descriptor al mismo tiempo para calcular las distancias entre objetos, entrega un mejor resultado que utilizar el mejor FV por separado. Esto, sin embargo, posee algunas limitaciones: No todas las combinaciones son igualmente buenas, y existen algunas que pueden incluso empeorar el resultado de utilizar FV's individuales. También existe un número óptimo de FV's a combinar, ya que cada vector agregado aumenta la efectividad de la consulta, pero esta mejora disminuye mientras más vectores se utilizan, llegando a un punto en que no conviene agregar más vectores.

Para comprobar este resultado en nuestro ambiente de pruebas, así como para determinar su comportamiento en el caso particular de la *PSB*, realizamos un experimento similar al anterior: Calculamos el *R-Precision* global para todas las combinaciones posibles de k FV's, con k variando entre 2 y el número total de vectores disponibles (11 en nuestro caso). Con esto podremos obtener un ranking sobre las mejores combinaciones para cada valor de k, y observar la mejora en comparación con los vectores individuales. Sin embargo, el cálculo de este ranking mediante fuerza bruta aumenta exponencialmente con la cantidad de FV's a combinar (en el caso de la PSB son necesarios 2^{11} experimentos distintos), y generalmente es costoso de calcular en otra MMDB más grande o con más FV's disponibles. La Tabla 4.6 muestra un ranking con la mejor combinación por cada valor de k, mientras que la Figura 4.2 ilustra el

N°	Nombre FV	R-Precision
1	Depth buffer	0.3116
2	Voxel	0.3113
3	Silhouette	0.2814
4	Complex	0.2707
5	3DDFT	0.2508
6	GRAY	0.2253
7	Rotation invariant	0.2252
8	Harmonics 3D	0.2015
9	Shape distribution	0.1826
10	Cords based	0.1574
11	Moments	0.1481

Cuadro 4.1: Ranking según desempeño para cada FV, en entorno de pruebas local.

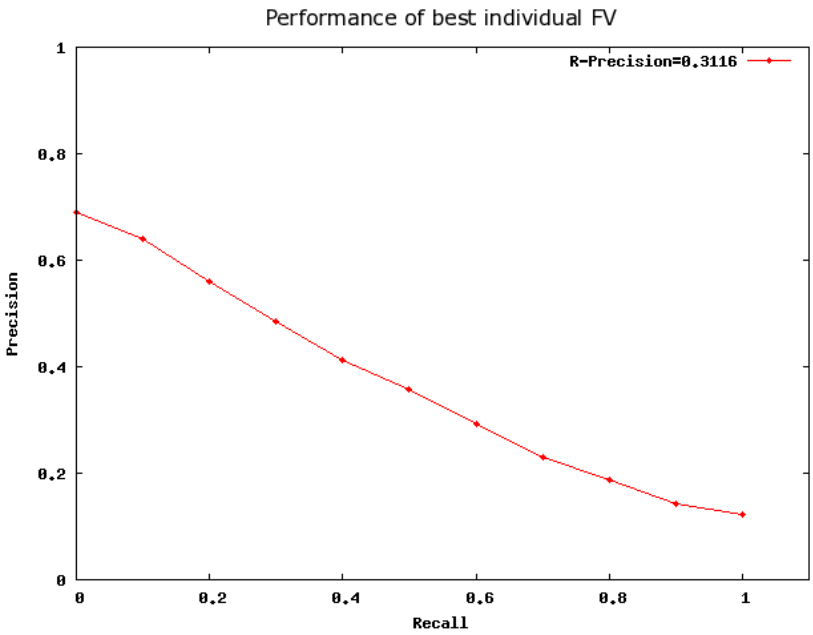

Figura 4.1: Diagrama *Precision v/s Recall* para el FV *Depth Buffer*, el mejor vector individual en promedio.

N°	Nombre FV	R-Precision	Posición en cuadro 4.1
1	Depth buffer	0.3220	1
2	Voxel	0.3026	2
3	Complex	0.2974	4
4	Rays-SH	0.2815	-
5	Silhouette	0.2736	3
6	3DDFT	0.2622	5
7	Shading-SH	0.2386	-
8	Ray based	0.2331	-
9	Rotation Invariant	0.2265	7
10	Harmonics 3D	0.2219	8
11	Shape Distribution	0.1930	9
12	Ray Moments	0.1922	-
13	Cords based	0.1728	10
14	Moments	0.1648	11
15	Volume	0.1443	-
16	Shape Spectrum	0.1119	-

Cuadro 4.2: Ranking según desempeño para cada FV, realizado en [1].

diagrama *Precision v/s Recall* para la mejor combinación de vectores a nivel general, la cual se obtuvo con $k = 4$. Se observó un resultado también conocido empíricamente, donde la mejor combinación de k vectores generalmente es igual a la mejor combinación de $(k-1)$ vectores, a la cual se le agrega un vector nuevo.

Para observar el comportamiento general de todas las combinaciones posibles de vectores, además de guardar la mejor combinación para cada valor de k, también se almacenaron los valores de *R-Precision* para cada combinación de vectores, con k desde 1 hasta 11. Esto genera un conjunto de pares (k, v), donde k es el número de vectores combinados, y v es el valor de *R-Precision* para una combinación dada. Al graficar dichos puntos, se observó que los valores de *R-Precision* se distribuyen uniformemente entre un valor mínimo y máximo, y el valor máximo se mantiene prácticamente constante a partir de cierto número de combinaciones. El valor mínimo, sin embargo, mejora mientras más vectores utilizamos. Al registrar las mejores combinaciones para cada valor de k, se observó que existen grupos de descriptores que funcionan mejor siendo combinados entre sí, en comparación a otros grupos. Esto se debe principalmente a la naturaleza de los descriptores al extraer información de los objetos multimedia. Si se combinan los mejores descriptores, se obtendrá rápidamente información precisa sobre el objeto,

N°	Nombre FV	R-Precision
1	Shape distribution	0.2380
2	Rotation invariant	0.2142
3	GRAY	0.2142
4	Depth buffer	0.1904
5	Complex	0.1428
6	Silhouette	0.1428
7	3DDFT	0.1190
8	Voxel	0.0952
9	Harmonics 3D	0.0952
10	Cords based	0.0476
11	Moments	0.0476

Cuadro 4.3: Ranking según desempeño de FV's para la clase 'Train'. Notar que *Shape distribution*, un FV que tiene mal desempeño global, es el mejor para esta clase, mientras que *Voxel*, uno de los mejores FV's globales, no se comporta tan bien aquí.

N°	Nombre FV	R-Precision
1	Depth buffer	0.5510
2	Rotation invariant	0.5351
3	Silhouette	0.5273
4	Voxel	0.5236
5	Complex	0.5069
6	Harmonics 3D	0.4151
7	Shape distribution	0.3963
8	3DDFT	0.3787
9	GRAY	0.3057
10	Cords based	0.2857
11	Moments	0.2408

Cuadro 4.4: Ranking según desempeño de FV's para la clase 'Human'. Se puede observar que todos los FV's presentan una mejora significativa en relación a su desempeño global.

N°	Nombre FV	R-Precision
1	Voxel	0.2450
2	Harmonics 3D	0.1950
3	GRAY	0.1850
4	Depth buffer	0.1849
5	3DDFT	0.1683
6	Silhouette	0.1466
7	Rotation invariant	0.1466
8	Complex	0.1000
9	Shape distribution	0.0899
10	Cords based	0.0633
11	Moments	0.0549

Cuadro 4.5: Ranking según desempeño de FV's para la clase 'Potted plant'. A pesar de que éste es bastante similar al ranking global, todos presentan un bajo valor de *R-Precision*, es decir, objetos de esta clase son difíciles de recuperar correctamente sin importar el FV utilizado.

sin embargo, si agregamos más vectores, la información aportada será menor en comparación a los primeros. De igual forma, si agregamos descriptores buenos a combinaciones malas, la información aumenta notablemente, lo que explica que el valor mínimo de *R-Precision* aumente mientras más vectores utilizamos. Las Figuras 4.3 y 4.4 ilustran este fenómeno con las combinaciones posibles de vectores en los conjuntos de entrenamiento y prueba en la base de datos *PSB*.

Previo al estudio propiamente tal de los conjuntos de entrenamiento, estimaremos el mejor resultado que podríamos obtener si tuviéramos conocimiento a priori sobre qué FV's nos conviene utilizar ante cada consulta. Esto tiene por objetivo determinar si es posible mejorar el desempeño de las consultas utilizando información *online* en lugar de la información estática sobre el comportamiento de cada FV.

Al calcular los mejores FV's a modo general, también fue posible determinar los mejores FV's por cada clase (observando la clase del objeto de consulta). Esta es la información contenida en las Tablas 4.3, 4.4 y 4.5. Además, para cada objeto mantenemos el mejor valor de *R-Precision* obtenido con algún FV. Esto se realiza en el conjunto de entrenamiento, con lo cual obtenemos los resultados que se indican en las Tablas 4.7 y 4.8. Obviamente el resultado mejora utilizando el FV adecuado ante cada situación, en vez de escoger uno fijo para todas las consultas. También se observa que hasta 5 ó 6 descriptores el valor del *R-Precision* se mantiene estable, lo cual es consistente con el resultado de las Figuras 4.3 y 4.4

k	Mejor combinación	R-Precision
1	Depth buffer	0.3116
2	Depth buffer, Voxel	0.3854
3	Depth buffer, Voxel, Complex	0.4063
4	Depth buffer, Voxel, Complex, Silhouette	0.4174
5	Depth buffer, Voxel, Complex, Silhouette, Moments	0.4127
6	Depth buffer, Voxel, Complex, Silhouette, Moments, GRAY	0.4003

Cuadro 4.6: Mejores combinaciones de FV's y sus respectivos *R-Precision*. En general se observa que cada vez que k se incrementa, un vector extra se agrega a la mejor combinación existente.

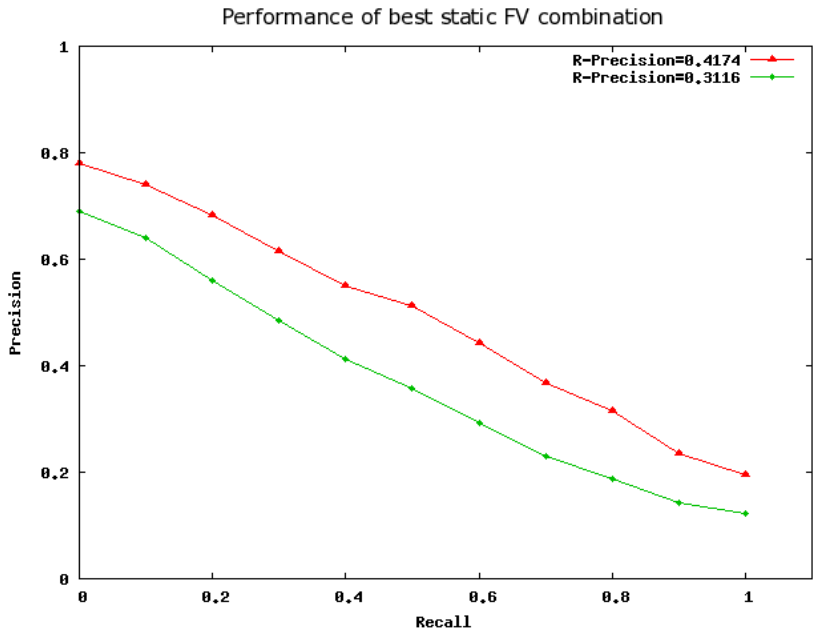

Figura 4.2: Diagrama *Precision v/s Recall* para la mejor combinación de vectores, lograda con los FV's *Depth Buffer*, *Complex*, *Voxel* y *Silhouette*. La curva inferior corresponde al mejor FV individual.

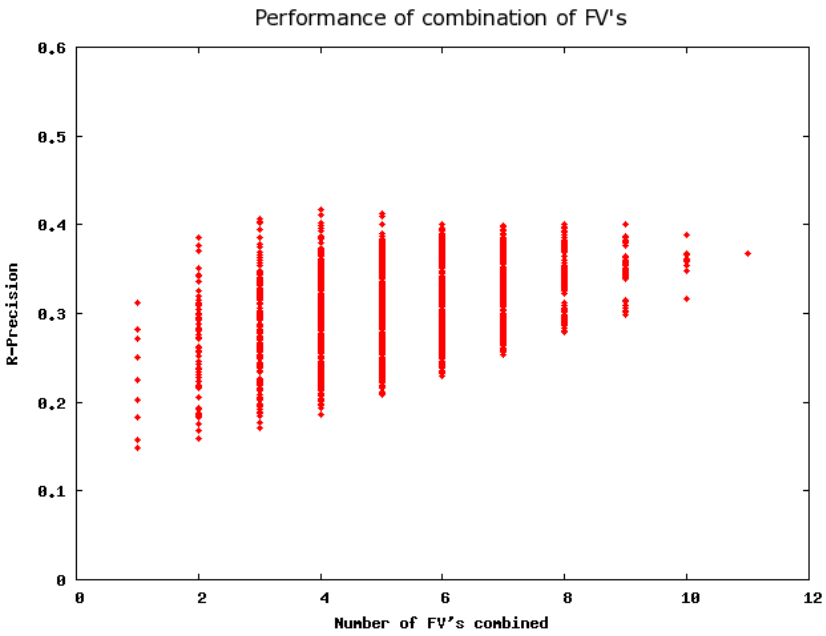

Figura 4.3: Valores de *R-Precision* para todas las combinaciones de k FV's, realizadas en el conjunto de entrenamiento de la *PSB*. Las combinaciones se distribuyen uniformemente entre un valor mínimo y uno máximo. También se observa que existe un valor óptimo de k, a partir del cual el *R-Precision* no mejora si se agregan más FV's.

4.3. Cálculo de los pesos para cada consulta

Para definir los pesos asignados a cada FV en una búsqueda por similitud, el primer paso consiste en verificar si el cálculo del indicador *entropy impurity* en cada consulta es consistente con el desempeño global y por clase de cada FV, encontrado en los experimentos exploratorios. Al mismo tiempo, se pretende determinar la robustez del resultado mismo ante varias ejecuciones del experimento.

Ya que el cálculo de *entropy impurity* requiere que todas las clases sean del mismo tamaño (de esta forma se evita que una clase tenga más probabilidad de aparecer que otra, lo cual puede sesgar el resultado), para este experimento mantendremos todas las clases que poseen 6 o más objetos en el conjunto de entrenamiento. En el caso de la *PSB*, esto implica mantener aproximadamente 60 de las 91 clases existentes. A su vez, de cada una de estas clases, seleccionaremos 3 objetos aleatoriamente, formando una *ground truth* de tamaño reducido para calcular el *entropy impurity* para cada consulta. La razón de los valores escogidos es que con 6 objetos podemos mantener una buena parte de las clases iniciales, y al mismo tiempo ser

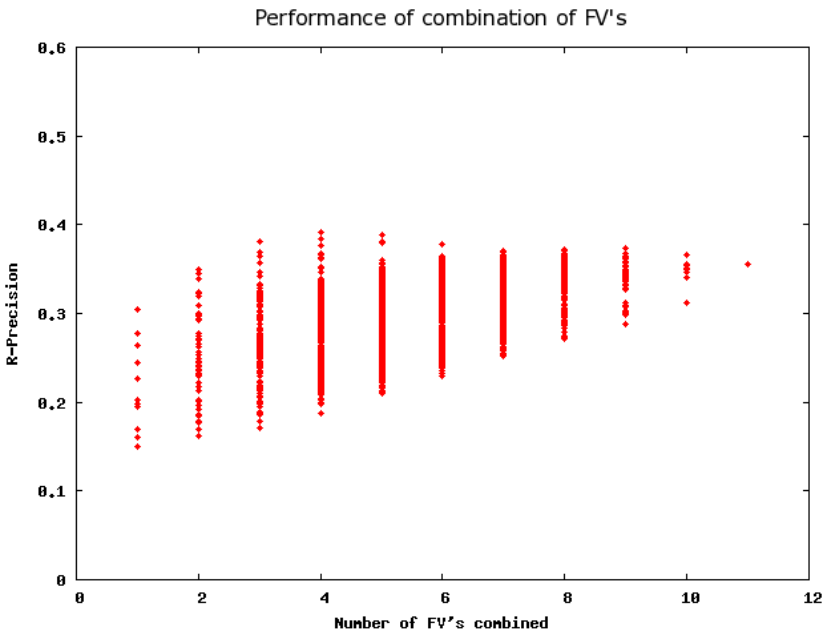

Figura 4.4: Valores de *R-Precision* para todas las combinaciones de k FV's, realizadas en el conjunto de prueba de la *PSB*. Aunque la distribución es uniforme entre el valor mínimo y el máximo, en general las mejores combinaciones poseen varios descriptores en común.

capaces de realizar varios experimentos con selecciones aleatorias de 3 objetos en cada ocasión. Para cada consulta, entonces, el valor de la *entropy impurity* puede tomar 3 valores distintos, dependiendo de las clases de los objetos retornados (todas iguales, todas distintas, o dos iguales y una distinta).

Dado que se busca construir una base de entrenamiento distinta de la MMDB original, utilizaremos los objetos del conjunto de prueba para realizar las consultas. Un objeto de consulta utilizará el conjunto de entrenamiento únicamente para determinar la coherencia de la respuesta entregada por cada FV, para posteriormente asignarle un peso a cada uno al realizar la búsqueda en la MMDB real.

Al calcular el *entropy impurity* utilizando el conjunto de prueba para seleccionar los objetos de consulta, y luego calculando un promedio sobre todas las consultas hechas, se obtienen los valores que se indican en la Tabla 4.9. Como también nos interesa la estabilidad de dichos valores a lo largo de cada experimento, también se indica el valor de la varianza obtenida a lo largo de éstos.

k	*R-Precision*
1	0.4232
2	0.4819
3	0.4975
4	0.4999
5	0.4963
6	0.4889
7	0.4751
8	0.4657
9	0.4422
10	0.4173
11	0.3673

Cuadro 4.7: *R-Precision* promedio si se utiliza la mejor combinación de k vectores para cada clase de objetos.

k	*R-Precision*
1	0.5022
2	0.5756
3	0.5948
4	0.5962
5	0.5888
6	0.5740

Cuadro 4.8: *R-Precision* promedio, utilizando el FV de mejor desempeño para cada objeto.

Se observa que el desempeño de los FV's de acuerdo a este indicador posee casi el mismo orden que el obtenido con el indicador *R-Precision* (ver Tabla 4.1). También es posible analizar el comportamiento por clases específicas, los cuales muestran comportamientos similares a los de las Tablas 4.3, 4.4 y 4.5. En las Tablas 4.10 y 4.11 se muestran algunos ejemplos de este comportamiento, indicando que el determinar la calidad de un FV utilizando el indicador *R-Precision* o el *entropy impurity* debería entregar resultados bastante similares.

Un resultado interesante es que el desempeño de un descriptor mediante el uso del *R-Precision* promedio varía linealmente con el *entropy impurity* obtenido en todas las consultas hechas. En la Figura 4.5 se muestra la relación entre estas variables, utilizando regresión lineal simple. También se incluye el valor de la correlación entre las variables, el cual resultó ser

N°	FV	*entropy impurity* (Avg.)	Var.
1	Depth buffer	0.8247	0.1024
2	Silhouette	0.8377	0.0969
3	Voxel	0.8766	0.0899
4	Complex	0.8830	0.0800
5	GRAY	0.9217	0.0691
6	Rotation invariant	0.9242	0.0679
7	3DDFT	0.9311	0.0671
8	Shape distribution	0.9424	0.0615
9	Harmonics 3D	0.9464	0.0555
10	Moments	0.9703	0.0501
11	Cords based	0.9797	0.0464

Cuadro 4.9: *Entropy impurity* promedio.

bastante alto. Este resultado era desconocido hasta ahora, por lo tanto puede ser considerado como un nuevo aporte al área de las MMDBs.

Finalmente, se comprobó que este resultado es prácticamente invariante ante el número de iteraciones utilizadas para calcular el *entropy impurity* promedio para cada FV. En las Figuras 4.6 y 4.7 se muestran ejemplos de la invariancia de este valor al realizar un número de iteraciones entre 5 y 50. Esto significa que necesitamos un número bajo de iteraciones para estimar con bastante certeza la calidad de un descriptor.

Hasta este punto, los experimentos han tenido como finalidad el repetir resultados ya conocidos en el área. En este sentido, todos los resultados han sido prácticamente iguales a los ya conocidos, por lo tanto se puede concluir que el entorno de pruebas es válido para continuar con la investigación de aquí en adelante.

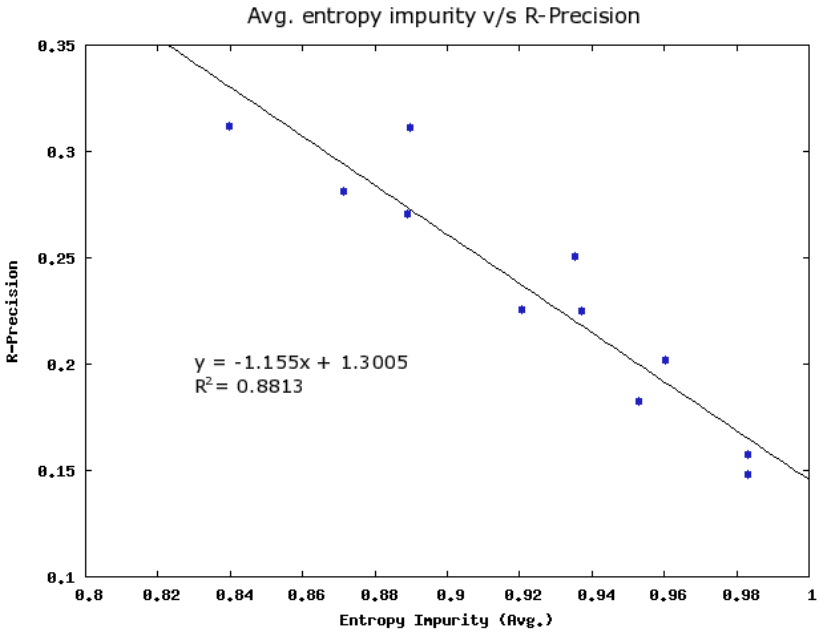

Figura 4.5: Correlación entre el *R-Precision* promedio y el *entropy impurity* obtenido al estimar la calidad de un FV.

N°	FV	*entropy impurity* (Avg.)	Var.
1	Silhouette	0.6236	0.1421
2	Depth buffer	0.6869	0.1522
3	Voxel	0.8037	0.1052
4	Complex	0.8053	0.0915
5	Rotation invariant	0.8331	0.0942
6	Shape distribution	0.8784	0.0617
7	Harmonics 3D	0.9026	0.0969
8	GRAY	0.9394	0.0566
9	3DDFT	0.9553	0.0569
10	Moments	0.9745	0.0503
11	Cords based	0.9859	0.0394

Cuadro 4.10: *Entropy impurity* promedio en la clase 'Human'.

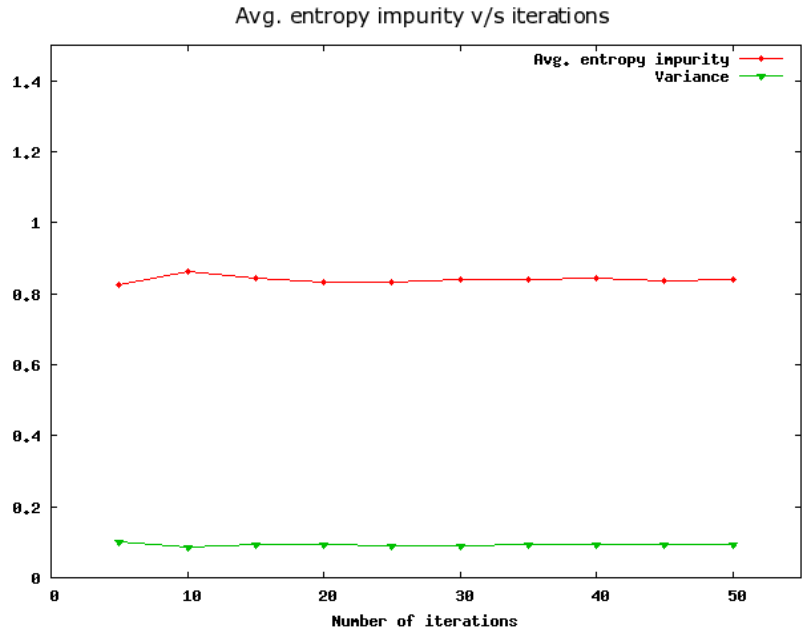

Figura 4.6: *Entropy Impurity* promedio y varianza obtenidas para el FV *Depth buffer* realizando un número variable de iteraciones para calcular dicho promedio.

N°	FV	*entropy impurity* (Avg.)	Var.
1	Silhouette	0.8813	0.0693
2	Depth buffer	0.9084	0.0732
3	Voxel	0.9387	0.0483
4	Harmonics 3D	0.9493	0.0467
5	3DDFT	0.9631	0.0604
6	GRAY	0.9658	0.0491
7	Rotation invariant	0.9671	0.0435
8	Shape distribution	0.9884	0.0388
9	Moments	0.9906	0.0436
10	Complex	1.0133	0.0321
11	Cords based	1.0275	0.0278

Cuadro 4.11: *Entropy impurity* promedio en la clase 'Potted plant'.

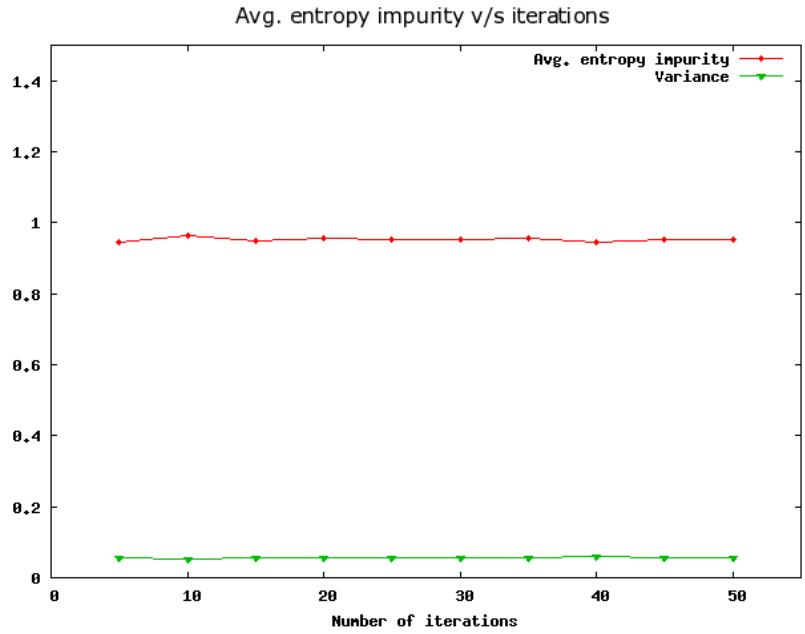

Figura 4.7: *Entropy Impurity* promedio y varianza obtenidas para el FV *Harmonics 3D* realizando un número variable de iteraciones para calcular dicho promedio.

Capítulo 5

Experimentos de investigación

En este capítulo comenzará la principal contribución de esta memoria. Mediante experimentos masivos, se buscará determinar la mejor forma de escoger un conjunto de entrenamiento, variando los parámetros para posteriormente realizar consultas con pesos calculados dinámicamente y mejorar la efectividad en una búsqueda por similitud.

Como ya ha sido mencionado, queremos encontrar alguna forma de construir un conjunto de entrenamiento de un cierto tamaño, dada una base de datos multimedia. Con este conjunto se pretende calcular los pesos de forma dinámica para cada FV, para luego realizar una búsqueda más adecuada según el objeto de consulta. En nuestro caso, queremos escoger un conjunto de entrenamiento como un subconjunto de la MMDB original, si es posible de un tamaño reducido para evitar que el cálculo de los pesos sea costoso en sí mismo.

Para estudiar este comportamiento, realizaremos un mismo experimento varias veces, en el cual se modificarán los siguientes parámetros para construir el conjunto de entrenamiento:

- *El número mínimo de objetos que debe poseer una clase para ser incluída.* La justificación consiste en que si una clase posee pocos objetos probablemente no alcance a ser definida correctamente, o incluso podría tratarse de objetos sin una clasificación concreta.

- *El número de objetos a escoger de cada clase.* Básicamente, una clase está definida por un número determinado de objetos. La pregunta es si existe un número óptimo de objetos a incluir para construir la *ground truth*, y si es así, determinar de qué forma este número afecta el rendimiento en una búsqueda por similitud.

- *La cantidad de clases a mantener en el conjunto de entrenamiento.* Si se busca determinar la calidad de un FV por la coherencia de las respuestas que entrega, es posible que con muy pocas clases las respuestas sean más coherentes de lo que serían en la práctica, o si son muchas, será más común retornar objetos de distintas clases. Por eso, resulta indispensable estudiar cómo varía la evaluación de un FV en relación a este parámetro.

Junto con todo esto, implícitamente estaremos observando si los resultados se repiten sistemáticamente en ejecuciones sucesivas del mismo experimento. De esta forma lograremos obtener resultados más robustos, los cuales podrán ser reproducidos posteriormente.

Para extraer un subconjunto de una MMDB existente, actualmente existen 3 tipos de filtros en el ambiente de pruebas, cada uno dedicado a filtrar un conjunto de objetos inicial según los parámetros mencionados. Estos filtros pueden ser combinados para lograr conjuntos más específicos, y nuevos filtros pueden ser agregados fácilmente a futuro.

De aquí en adelante, a menos que se indique lo contrario, la forma de evaluar la calidad de un conjunto de entrenamiento consiste en calcular los pesos asignados a cada FV para un cierto objeto de consulta, utilizando el método *entropy impurity*, y luego calcular el *R-Precision* para la consulta en la MMDB real. La ejecución de cada experimento consiste en seleccionar un objeto del conjunto de prueba, obtener los pesos para cada FV en el conjunto de entrenamiento (previamente filtrado), y luego ocupar los pesos obtenidos para realizar la consulta en el mismo conjunto de prueba, teniendo en cuenta que el objeto mismo no debe formar parte de la respuesta. La Figura 5.1 ilustra en detalle este esquema de ejecución de experimentos. Los resultados entregados en forma de tablas o gráficos serán el resultado de promediar los resultados individuales para todos los objetos de consulta.

Para cada combinación posible de las variables a estudiar se ha creado un gráfico que indica cómo se desempeñan en promedio las consultas por similitud utilizando los pesos calculados mediante *entropy impurity*. De esta manera, el estudio de cada variable se puede hacer tomando los gráficos en que las demás variables se mantienen constantes. Cada gráfico ilustra cómo se comporta el *R-Precision* en promedio a medida que aumentamos el número de clases en la *ground truth*. Además se incluirán curvas para comparar la consulta dinámica con 3 casos estáticos: El mejor descriptor individual, la combinación de todos los descriptores con pesos unitarios, y utilizar el mejor descriptor por objeto (una especie de oráculo).

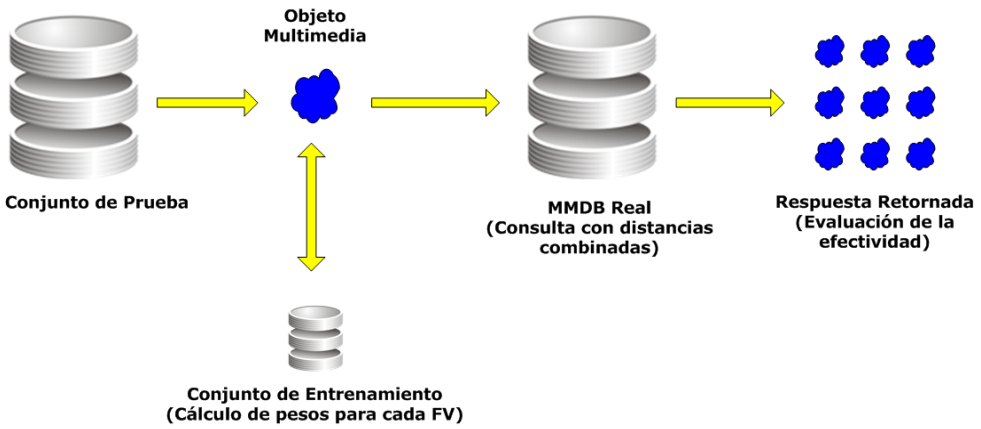

Figura 5.1: Esquema general de ejecución de un experimento. Para cada objeto de consulta seleccionado, se realizan consultas previas en un conjunto de entrenamiento para calcular los pesos de cada FV. Con estos pesos se realiza la consulta en la MMDB real, y la respuesta retornada es finalmente evaluada con las métricas correspondientes.

5.1. Efecto del número de objetos escogidos por clase

Al seleccionar las clases para nuestra *ground truth* de una cierta forma, lo ideal sería poder escoger un cierto número de objetos de cada una para formar un conjunto de entrenamiento de tamaño reducido en relación al tamaño de la MMDB original. En general, las clases seleccionadas para este propósito pueden tener una cantidad variable de objetos. A modo de ejemplo, en la PSB existen clases con muchos objetos, como las de figuras humanas, o clases más específicas, como tipos bien definidos de animales o medios de transporte. El escoger el mismo número de objetos por cada clase tiene como finalidad mantener un tamaño acotado del conjunto de entrenamiento, evitar que algunas clases tengan mayor probabilidad de obtener mejores pesos, y decidir cuáles son los objetos que vamos a escoger.

Para estudiar el comportamiento del número de objetos por clase, los gráficos son agrupados según esta variable, la cual toma valores entre 2 y 7 (con más de 7 objetos no hubo grandes cambios en los experimentos). El número de objetos por clase determina el rango de valores que podemos obtener en el cálculo del *entropy impurity* (a medida que se utilizan más objetos por clase la cantidad de valores distintos que puede tomar el *entropy impurity* también aumenta). Al agrupar los gráficos según esta variable, el siguiente comportamiento fue observado:

1. Al utilizar 2 objetos por clase, el desempeño general de las consultas fue inferior incluso a utilizar el mejor vector individual. Ya que los pesos pueden tomar 2 valores distintos, esto equivale a escoger o no un determinado FV para la combinación, lo cual es una medida demasiado gruesa para distinguir qué vectores son mejores que otros para una consulta dada.

2. Con 3 ó 4 objetos por clase, el desempeño general supera a escoger el mejor vector individual, y en ocasiones puede superar a la combinación estática de todos los descriptores. Esto se debe a que la cantidad de valores posibles para *entropy impurity* permite distinguir mejor qué vectores son mejores que otros.

3. A medida que cada clase posee más objetos, el desempeño empieza a ser cada vez más similar a combinar todos los vectores con pesos unitarios. Esto se debe a que una consulta que retorna más objetos tendrá más posibilidades de retornar objetos de clases distintas. Esto hace que el indicador *entropy impurity* entregue pesos muy parecidos para cada vector.

Este comportamiento se mantuvo relativamente constante al cambiar los valores de las demás variables. De la misma manera, el comportamiento observado al utilizar muchos objetos por clase se mantuvo prácticamente igual utilizando más de 6 ó 7 objetos por clase. La Figura 5.2 muestra el comportamiento general de las consultas para distintos valores de esta variable.

5.2. Efecto del tamaño mínimo de una clase en la *ground truth*

Para estudiar cómo afecta el tamaño mínimo de una clase en la *ground truth*, se agruparán los gráficos generados según el valor de esta variable. Para el caso de la PSB, un rango aceptable de valores se encuentra entre 3 y 8 objetos. Más allá de este valor es difícil hacer un análisis correcto del comportamiento, ya que la gran mayoría de las clases de la PSB quedarían descartadas, lo cual produciría gráficos con un rango muy estrecho de valores en el eje x (número de clases mantenidas). El estudiar esta variable permite saber si una clase para la *ground truth*

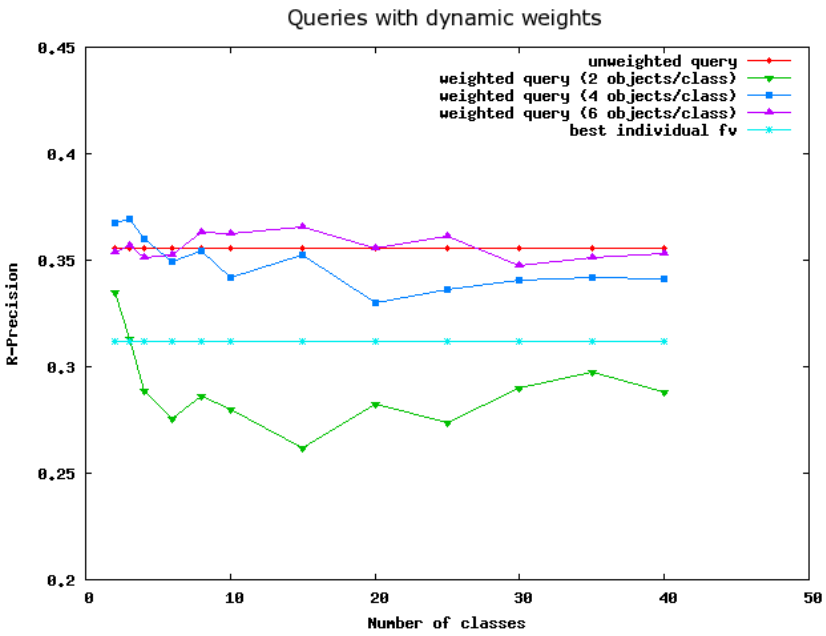

Figura 5.2: Desempeño general de consultas utilizando un número distinto de objetos por clase. El bajo desempeño con 2 objetos se debe a que es difícil saber con certeza qué vectores son mejores que otros observando sólo los dos primeros elementos retornados. Al aumentar este número a 6 ó más objetos, el desempeño se hace similar a combinar todos los vectores con pesos unitarios. El óptimo para el caso de la PSB se encuentra alrededor de 4 objetos, donde es posible mejorar el resultado obtenido mediante una consulta estática.

funciona mejor al ser muy numerosa (por ejemplo, figuras o imágenes humanas, fáciles de encontrar en MMDBs públicas), o sólo se necesita que tenga unos pocos elementos capaces de definirla bien.

Al obtener los gráficos respectivos, y compararlos según el valor del tamaño mínimo de una clase, se llegó a un resultado general: El cambio en el tamaño mínimo de una clase no tiene una influencia clara en el desempeño de una consulta a modo general. Cualquiera de los gráficos generados es prácticamente similar a otro que tenga los mismos valores de las demás variables y un tamaño mínimo distinto. La aleatoriedad producida entre un gráfico y otro se debe principalmente a las clases conservadas en la *ground truth*, las cuales son escogidas aleatoriamente.

Este resultado sugiere que una clase en la *ground truth* no necesita contener demasiados objetos para ser útil al evaluar un descriptor. Más que la cantidad, lo que se necesita es que los objetos seleccionados sean capaces de representar con certeza una clase o concepto dentro de los

objetos multimedia. Por ejemplo, si una clase sobre automóviles posee muy pocos objetos, aún así esta clase puede ser incluida en la *ground truth*, si dichos objetos son buenos representantes de la clase. Ejemplos de este comportamiento se observan en las Figuras 5.3 y 5.4.

5.3. Efecto de la cantidad de clases escogidas

Dada la forma de generar los gráficos, en todos ellos se visualiza el cambio en el desempeño al variar el número de clases presentes en la *ground truth*. Cada gráfico contiene en el eje x el número de clases mantenidas en la *ground truth* y en el eje y el *R-Precision* promedio obtenido utilizando todos los objetos de consulta. En todos ellos se pueden observar patrones de comportamiento similares, salvo en casos muy puntuales, donde probablemente las clases escogidas no hayan sido muy buenas (en todos los experimentos fueron escogidas en forma aleatoria).

1. Al utilizar un número reducido de clases, el desempeño general mejora mientras menos clases son utilizadas. Para observar con detalle el comportamiento en este rango, el número de clases es variado lentamente entre 2 y 10 clases. Aunque en este rango el valor de *R-Precision* puede variar bastante entre un experimento y otro, este valor tiende a aumentar, alcanzando su valor máximo generalmente utilizando 2 ó 3 clases. Esto indica que probablemente para identificar un buen descriptor sólo se necesita saber si es capaz de distinguir objetos de un conjunto muy pequeño de clases.

2. Cuando se aumenta el número de clases a más de 10, el valor promedio de *R-Precision* se mantiene prácticamente constante, pero en general por debajo del valor obtenido con menos clases. Esto se puede explicar porque al existir muchas clases, un descriptor que puede ser bueno identificando clases distintas no será capaz de funcionar de igual manera si las clases se encuentran muy cercanas entre sí, ya que esto provocará que el descriptor entregue rápidamente objetos de clases distintas (En altas dimensiones es muy frecuente que grupos de objetos se intersecten o se encuentren cerca entre sí).

Aunque los gráficos generados seguían diversos patrones según la combinación del resto de las variables, en todos ellos se observa una degradación del desempeño si se agregan muchas clases. El único caso en que esto no ocurre es cuando otra variable causa que la consulta se reduzca a utilizar todos los descriptores con pesos unitarios. Las variaciones entre un experimento y otro al utilizar clases se debe principalmente a las clases escogidas. Por ejemplo, si dos clases fueran suficientes para evaluar la calidad de un descriptor, el resultado podría seguir

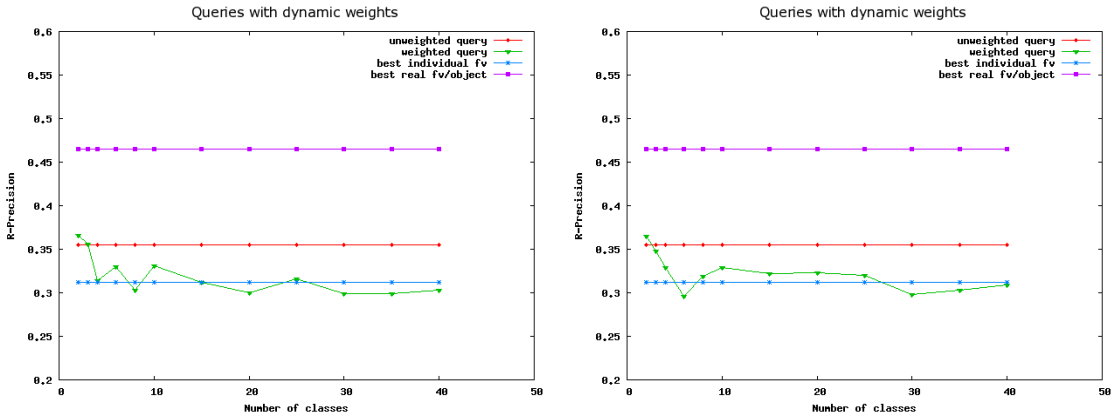

Figura 5.3: Desempeño general de consultas en base al tamaño mínimo de una clase para ser agregada a la *ground truth*. Teniendo en cuenta la variación causada por escoger clases aleatoriamente, el tamaño mínimo de una clase no parece ser determinante para construir un buen conjunto de entrenamiento. Los tamaños mínimos escogidos fueron de 3 objetos en el gráfico izquierdo, y de 8 objetos en el derecho.

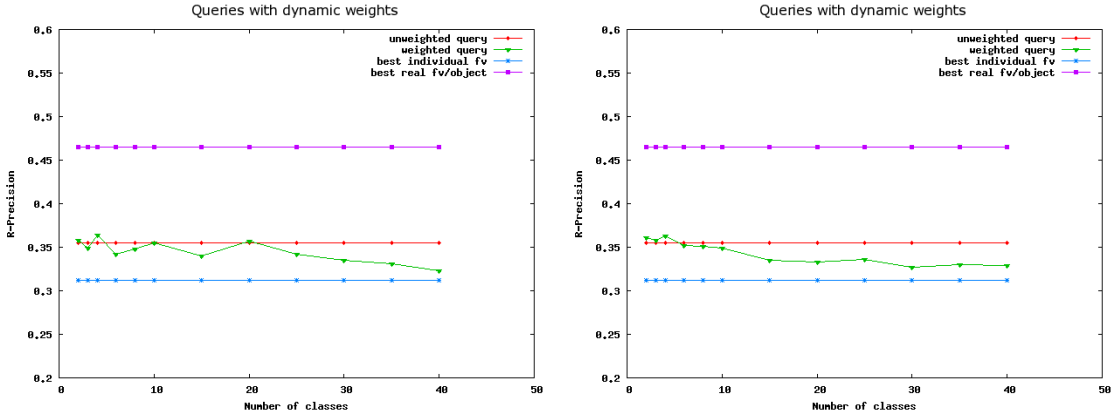

Figura 5.4: Desempeño general de consultas en base al tamaño mínimo de una clase. Los patrones detectados utilizando las otras variables se mantienen, sin importar el tamaño original de la clase. Se utilizó un tamaño mínimo de 4 objetos en el gráfico izquierdo, y de 6 objetos en el derecho.

siendo deficiente si se escogen clases muy cercanas entre sí. En cambio, si las clases son bien escogidas, esto reflejará de mejor manera la calidad de un descriptor al poder diferenciarlas. Las Figuras 5.5 y 5.6 ilustran el comportamiento del número de clases con distintos valores de las demás variables.

5.4. Análisis de los resultados

Después de realizar en forma masiva, y con un grado de aleatoriedad, todos los experimentos descritos en la sección anterior, se puede deducir que cada variable influye en alguna medida en una búsqueda por similitud, cuando ésta utiliza un conjunto de entrenamiento para calcular los pesos de cada descriptor asociado. De las variables estudiadas anteriormente, el tamaño mínimo de una clase prácticamente no tuvo ninguna influencia, lo que se pudo confirmar observando los gráficos correspondientes.

De igual forma, a modo general se pudo observar que el número de objetos en una clase posee un cierto valor óptimo. Con pocos objetos es difícil alcanzar a distinguir un buen descriptor, mientras que si se utilizan muchos, los pesos asignados a cada vector se empiezan a hacer prácticamente iguales, lo que degrada en utilizar todos los vectores con pesos unitarios. El utilizar muchos objetos impide que se puedan realizar mejoras mediante las otras variables, ya que de todas formas los pesos seguirán siendo parecidos entre cada vector.

Por otra parte, el número de clases claramente afecta el desempeño de una búsqueda por similitud. Con un número reducido de clases (que puede depender de la base de datos), la calidad de un descriptor puede ser evaluada mediante qué tan pura es la respuesta que entrega ante una cierta consulta. Si se utilizan muchas clases, este resultado se degrada rápidamente. Algunas razones de este comportamiento son la mayor probabilidad de que las clases estén muy cercanas entre sí, lo cual entregará una impureza más alta de la que entregaría un descriptor en condiciones normales, y la mayor cantidad de clases distintas en la respuestas obtenidas, ya que utilizando un número bajo de clases, el tipo de objetos distintos en la respuesta se mantendrá acotado por dicho número.

Siguiendo el comportamiento global de cada variable, conviene analizar con un poco más de detalle el número de clases y el número de objetos por clase. Estas variables fueron posteriormente analizadas en forma conjunta, dentro del rango de valores en que influyen en el conjunto

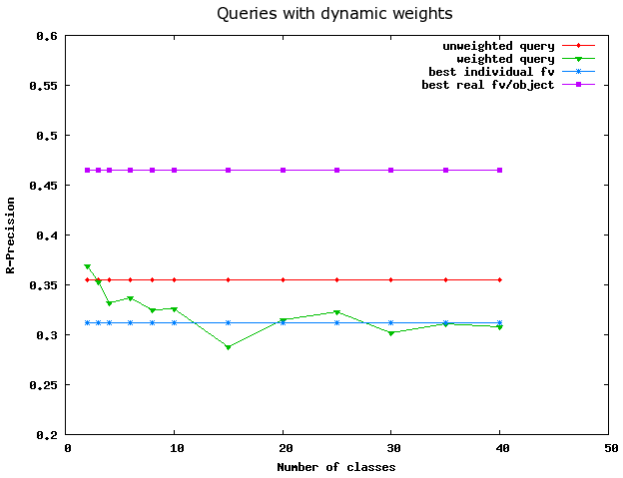

Figura 5.5: Desempeño general de consultas en base al número de clases en la *ground truth*. El comportamiento general observado es un valor alto del *R-Precision* utilizando pocas clases, el cual degrada rápidamente si se agregan más clases. La disminución en el *R-Precision* también depende de las demás variables. Este gráfico fue creado utilizando 4 objetos por clase y un tamaño mínimo de 6 objetos por clase.

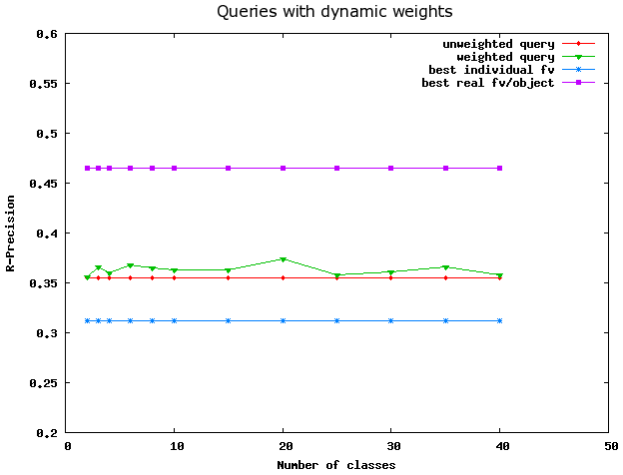

Figura 5.6: Desempeño general de consultas en base al número de clases. Se utilizaron 6 objetos por clase y un tamaño mínimo de 7 objetos por clase. Bajo esta configuración, el valor del *R-Precision* promedio no depende del número de clases en la *ground truth*.

de entrenamiento. Tomando los gráficos anteriores, se calculó el *R-Precision* promedio para cada combinación posible, utilizando todos los objetos del conjunto de prueba para calcular dicho promedio. Esto nos permite crear una matriz de valores para cada combinación, la cual se muestra en la Tabla 5.1.

	2	3	4	5	6	7
2	0.3250	0.3533	0.3583	0.3592	0.3572	0.3586
3	0.2979	0.3533	0.3543	0.3587	0.3590	0.3622
4	0.2836	0.3426	0.3547	0.3630	0.3587	0.3604
6	0.2833	0.3320	0.3552	0.3589	0.3624	0.3591
8	0.2780	0.3263	0.3508	0.3548	0.3595	0.3643
10	0.2720	0.3232	0.3467	0.3558	0.3576	0.3588
15	0.2748	0.3180	0.3422	0.3544	0.3592	0.3629
20	0.2812	0.3090	0.3360	0.3485	0.3562	0.3622
25	0.2801	0.3089	0.3332	0.3522	0.3574	0.3586
30	0.2843	0.3368	0.3368	0.3492	0.3553	0.3596
35	0.2927	0.3046	0.3321	0.3467	0.3556	0.3621
40	0.2897	0.3061	0.3307	0.3458	0.3551	0.3590

Cuadro 5.1: *R-Precision* promedio obtenido con todas las combinaciones posibles de número de clases y objetos por cada clase. Las filas indican el número de clases mantenidas en la *ground truth*, y las columnas indican la cantidad de objetos escogidos por cada clase.

Para una mejor comprensión de cómo se comportan estas variables, se crearon 2 gráficos, uno para cada variable, en los cuales se graficaron los puntos respectivos para cada valor de la misma, y una curva indicando el promedio respectivo por cada valor. Las Figuras 5.7 y 5.8 ilustran la tabla anterior utilizando una variable a la vez.

Al analizar la distribución de los valores según el número de clases en la *ground truth*, se observa que con pocas clases el *R-Precision* promedio siempre se mantiene en un valor alto, mientras que con más clases este indicador varía mucho más. Esto explica el comportamiento de los gráficos de las secciones anteriores, cuyo *R-Precision* promedio siempre era menor al utilizar más clases.

Por otra parte, al observar la variación según el número de objetos por clase, se puede comprobar que el rendimiento en promedio aumenta mientras más objetos se utilizan en la ground truth. Como se observó en la Figura 5.2, el utilizar muchos objetos por clase empieza

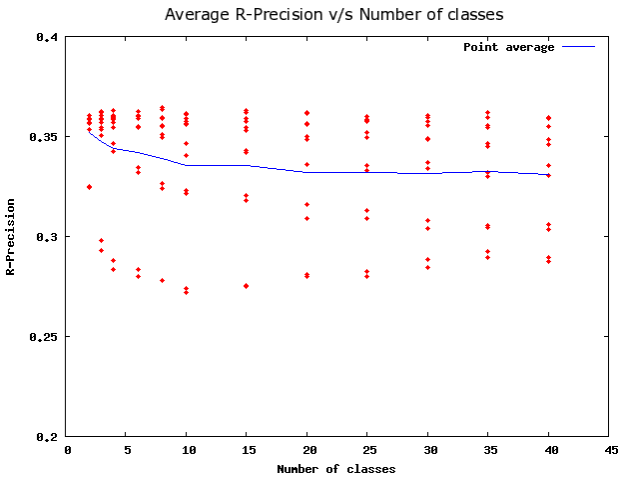

Figura 5.7: Evaluación general, vista desde la perspectiva del número de clases en la *ground truth*. El promedio aumenta levemente al utilizar más objetos por clase, al mismo tiempo que los valores empiezan a tener menor varianza.

a tener el mismo desempeño que combinar todos los vectores en forma estática. Sin embargo, esto limita las posibilidades de obtener un resultado mejor cambiando el valor de las demás variables. Con un número menor de objetos el promedio disminuye, pero los valores más altos se mantienen constantes.

Volviendo a la Tabla 5.1, existen dos formas de combinar estas variables para obtener valores altos del *R-Precision* promedio: Mantener un número bajo de clases o un número alto de objetos por clase. Dependiendo de la MMDB escogida y posiblemente de los FV existentes para el tipo de objetos en ella, la combinación óptima podría estar en un punto distinto de este conjunto. La alternativa que ofrece mayores posibilidades a priori de mejorar el desempeño es mantener un número bajo de clases, ya que éstas se podrían elegir de manera inteligente para mejorar la construcción del conjunto de entrenamiento, en cambio, utilizar muchos objetos por clase tiene las limitaciones de no poder mejorar más el caso de la combinación estática.

Para comprobar si efectivamente las clases escogidas influyen en el resultado, se realizó un último experimento, en el cual se construyeron conjuntos de entrenamiento de tamaño fijo, con sólo dos clases y 4 objetos por clase, utilizando todas las combinaciones posibles de 2 clases. Con cada conjunto se realizó el mismo experimento general: Calcular el *R-Precision* promedio obtenido mediante consultas con todos los objetos del conjunto de prueba. Con esta informa-

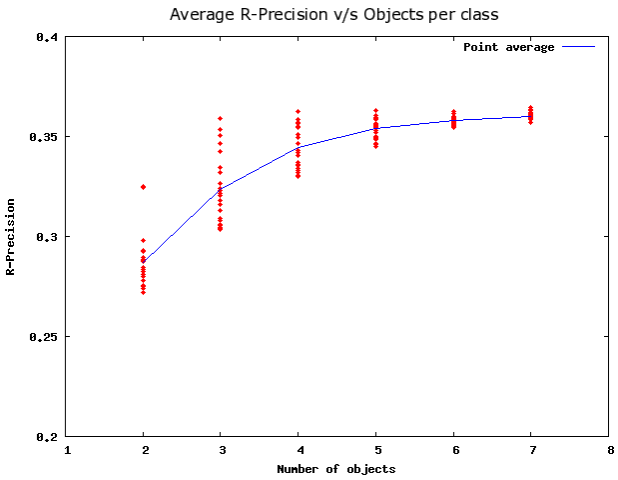

Figura 5.8: Evaluación general, vista desde la perspectiva del número de objetos por clase en la *ground truth*. Claramente el mantener más objetos mejora el valor del *R-Precision* promedio, al mismo tiempo que reduce la varianza de los valores obtenidos. Sin embargo, esto puede ser una limitación al querer mejorar el caso estático de combinar todos los vectores con el mismo peso.

ción fue posible generar un ranking de las combinaciones que lograban un mejor rendimiento, y el resultado observado fue que existe un cierto número de clases que aparecen con frecuencia en las mejores combinaciones posibles. Esto sugiere que las clases a mantener en el conjunto de entrenamiento no deberían ser escogidas aleatoriamente, sino que existe un cierto número de ellas que funcionan mejor al ser combinadas entre sí. Teniendo en cuenta esta información, el valor del *R-Precision* promedio puede aumentar ligeramente, con la ventaja de que su valor permanecerá alto si el grupo de clases se mantiene fijo, en lugar de la variación observada en los experimentos anteriores. Dado que esta memoria está enfocada en el tamaño del conjunto de entrenamiento, no se hizo un estudio exhaustivo de cómo elegir las clases de manera apropiada, aunque este último experimento indica que el desempeño también depende de las clases escogidas en el conjunto de entrenamiento.

Para concluir, al fijar el conjunto de entrenamiento de la PSB a 2 clases con 4 objetos cada una, se obtuvo el diagrama *Precision v/s Recall* de la Figura 5.9, la cual empieza a acercarse bastante a la mejor combinación general de descriptores, obtenida utilizando 4 de ellos. Esto indica que es posible aproximarse a casos desconocidos utilizando únicamente información *online*. En el caso de la PSB, la mejor combinación ya era conocida de antemano, pero si en otra MMDB no se tiene esta informacíón, es posible aproximarla con un conjunto de entrenamiento

bien escogido. Esta mejora podría ser incluso mayor en otras bases de datos, ya que en la PSB el asignar pesos a los vectores aumentó sólo levemente el desempeño en comparación a utilizar combinaciones estáticas de los vectores.

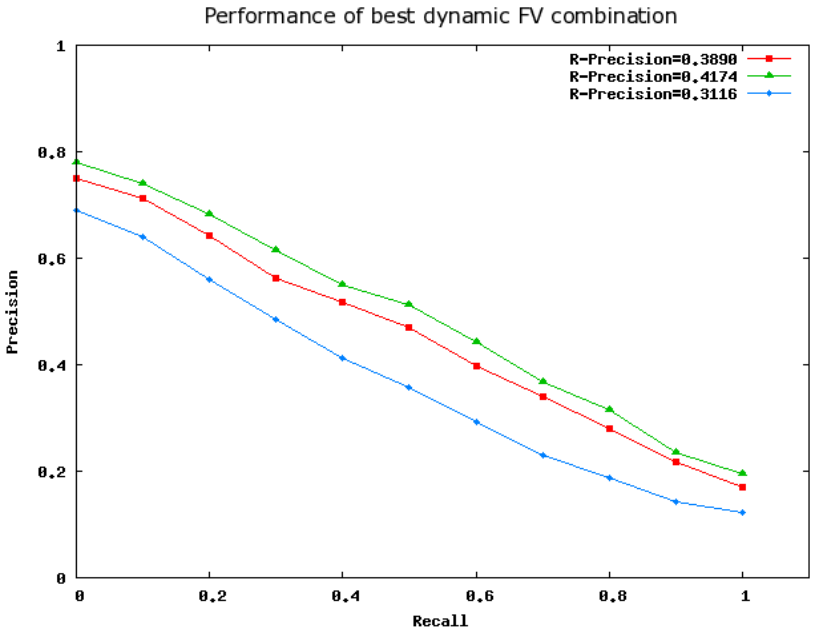

Figura 5.9: Diagrama *Precision v/s Recall* para el mejor tamaño del conjunto de entrenamiento, logrado con 2 clases y 4 objetos por clase (curva intermedia). Se observa que el cálculo hecho en forma dinámica tiende a acercarse a la mejor combinación estática (curva superior), calculada mediante fuerza bruta. Además se incluye el mejor vector individual (curva inferior).

5.5. Resultados en una base de datos de imágenes

Para comprobar si los resultados obtenidos en la parte experimental son independientes de la base de datos escogida, durante la ejecución de los experimentos en la PSB se empezó a trabajar en paralelo con otra base de datos, compuesta por aproximadamente un millón de imágenes. Aunque el utilizar sólo dos bases de datos distintas no es suficiente para deducir resultados globales aplicables a todas las bases de datos, es posible tener una noción más precisa sobre qué resultados deberían permanecer invariantes.

Este conjunto posee algunas diferencias en comparación a la PSB: Cada objeto posee 5 representaciones como FV, en lugar de las 11 existentes en la PSB. Además las imágenes no

poseen clasificación alguna, por lo tanto, antes de utilizarla, se tuvo que hacer una clasificación manual de un subconjunto de ellas. Debido al tiempo y espacio computacional empleado por los experimentos, este conjunto de imágenes fue reducido a unos 2.500 objetos clasificados, siguiendo la misma idea de la PSB en mantener los conjuntos de entrenamiento y prueba equilibrados con la misma cantidad y tamaño de las clases.

Al clasificar un subconjunto de las imágenes disponibles, se presentan problemas distintos a los existentes en objetos 3D. Por ejemplo, una imagen puede tener más de un concepto asociado con ella, a diferencia de los objetos 3D, los cuales modelan en general un sólo objeto del mundo real. Por ejemplo, una imagen donde aparece una persona en un bosque podría ser clasificada como una imagen de persona o una de paisaje natural, y no hay una forma clara de decidir cuál de las dos elegir. También existe el problema de decidir qué clases crear en la base de datos, dado que no se conoce la totalidad de las imágenes existentes. Por esta razón, los resultados presentados en esta sección poseen valores más bajos que los obtenidos en la PSB.

Una vez definida la base de datos de imágenes, inicialmente se calculó el desempeño individual de cada FV para construir el ranking apropiado, y luego comprobar los comportamientos al combinar varios de ellos y al calcular el indicador *entropy impurity* en forma separada. El ranking de desempeño se muestra en la Tabla 5.2, el cual, como se puede observar, posee valores de *R-Precision* más bajos que los observados en la PSB. Además, en la Tabla 5.4 y la Figura 5.11 se comprueban la relación entre la calidad de un FV y el valor promedio del *entropy impurity*, así como la correlación entre este valor y el *R-Precision* obtenido.

N°	Nombre FV	R-Precision
1	Color Structure	0.1450
2	Scalable Color	0.1371
3	Edge Histogram	0.1363
4	Color Layout	0.1278
5	Homogeneous Texture	0.1101

Cuadro 5.2: Ranking según desempeño para cada FV, en la base de datos de imágenes.

Luego de este cálculo, a continuación se comprobó que combinar varios de ellos a la vez mejora el valor del *R-Precision* promedio, aunque a diferencia de la PSB, el agregar más vectores no disminuye el rendimiento (excepto al aumentar de 4 a 5 vectores). Esto se puede

k	Mejor combinación	R-Precision
1	CS	0.1450
2	CS, EH	0.1685
3	CS, EH, CL	0.1761
4	CS, EH, CL, SC	0.1767
5	CS, EH, CL, SC, HT	0.1743

Cuadro 5.3: Mejores combinaciones de FV's y sus respectivos *R-Precision* en la base de datos de imágenes. CS: Color Structure, EH: Edge Histogram, CL: Color Layout, SC: Scalable Color, HT: Homogeneous Texture.

N°	FV	*entropy impurity* (Avg.)	Var.
1	Color Structure	1.0163	0.0197
2	Color Layout	1.0321	0.0311
3	Edge Histogram	1.0353	0.0274
4	Scalable Color	1.0377	0.0266
5	Homogeneous Texture	1.0534	0.0197

Cuadro 5.4: *Entropy impurity* promedio para los FV's de la base de datos de imágenes.

explicar porque con pocos vectores a combinar no se alcanza a observar el efecto de saturación entre ellos. El desempeño de las mejores combinaciones de k vectores, así como los valores del *R-Precision* promedio con todas las combinaciones posibles se muestran en la Tabla 5.3 y la Figura 5.10, respectivamente.

Luego de obtener resultados muy similares comparados a los obtenidos en la PSB, es muy probable que el comportamiento utilizando distintos conjuntos de entrenamiento sea similar. Como el tamaño de esta base de datos es bastante parecida a la PSB, se utilizaron los mismos rangos de valores para cada variable, obteniendo los siguientes resultados generales para cada una:

1. Según el número de objetos por clase, el comportamiento fue muy similar al de la PSB. Con un número reducido de objetos por clase el rendimiento es inferior incluso al mejor vector individual, pero a medida que se agregan más objetos el *R-Precision* promedio aumenta. Cerca de los 4 objetos el mejor valor del *R-Precision* puede alcanzar a la combinación estática de los vectores (utilizando pocas clases), mientras que si se agregan más objetos la curva completa se acerca a este valor. La única diferencia con respecto a

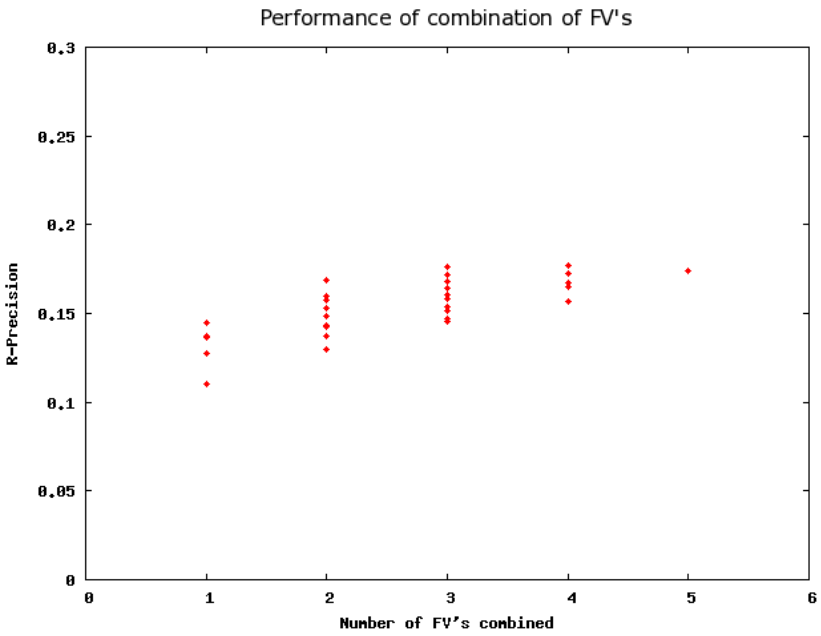

Figura 5.10: Valores de *R-Precision* para todas las combinaciones de k FV's, calculado en el conjunto de prueba en la base de datos de imágenes.

la PSB es que con 2 objetos por clase el *R-Precision* promedio fue bastante más bajo. En la Figura 5.12 se observa el comportamiento en paralelo de ambas bases de datos.

2. El tamaño mínimo de una clase tampoco influye en el resultado obtenido mediante el conjunto de entrenamiento. Los gráficos generados en cualquier rango de valores seguían los mismos patrones sin importar qué tamaño mínimo se exigiera a una clase para ser incluida. Ya que este resultado permaneció prácticamente constante en todos los gráficos de ambas bases de datos, se podría concluir con gran seguridad que, independiente de la base de datos escogida, el tamaño mínimo de una clase es irrelevante para el conjunto de entrenamiento.

3. El número de clases en el conjunto de entrenamiento también debe ser mantenido en un valor bajo para mejorar el desempeño de las consultas. En la mayoría de los casos estudiados, los gráficos tienden a presentar valores altos del *R-Precision* con pocas clases, disminuyendo este valor a medida que se agregan más clases. Sin embargo, hay un caso en que sucede lo contrario: Utilizando 2 objetos por clase, el desempeño aumenta a medida

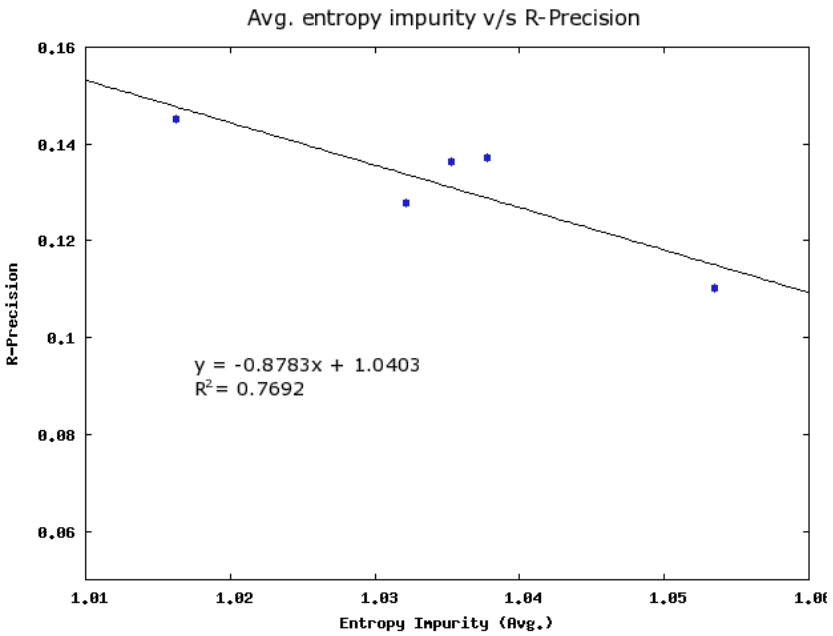

Figura 5.11: Correlación entre el *R-Precision* promedio y la varianza obtenida para los FV's de la base de datos de imágenes.

que se agregan más clases. Una explicación para este resultado es que la *ground truth*, por las razones explicadas, no está muy bien definida. Esto hace que los pesos no reflejen claramente la calidad de cada FV, asignando un peso nulo o un cierto valor positivo fijo, lo cual es aproximadamente igual a escoger sólo algunos FV's para la combinación, lo cual no es una buena elección según la Figura 5.10. En cambio, con pesos parecidos entre todos los FV's la consulta se acerca a combinar todos los vectores, que de acuerdo a la misma figura es uno de los mejores casos estáticos. De todas formas, mejores resultados se obtienen con clases con 4 objetos, y en dicho caso las curvas demuestran las mismas tendencias que las obtenidas en los experimentos con la PSB. Las Figuras 5.13 y 5.14 ilustran los comportamientos de ambas bases de datos en dos casos distintos.

Por último, se estudió el comportamiento del número de clases y de objetos por clase en forma conjunta, llegando a resultados muy similares a los obtenidos en la PSB, en las cuales se pueden obtener valores altos de *R-Precision* utilizando ya sea muchos objetos por clase ó pocas clases (ver Figuras 5.7 y 5.8). Los gráficos fueron ligeramente distintos debido a las diferencias puntuales señaladas anteriormente, sin embargo, el comportamiento global en ambas bases de datos tiende a ser el mismo. En la Figura 5.15 se muestra el resultado de agrupar los valores

de *R-Precision* según cada variable.

De la comparación general de ambas bases de datos, se pueden deducir comportamientos similares, los que probablemente no dependen de la base de datos escogida. Ejemplos de esto son, por ejemplo, el rango de valores óptimo para mejorar el desempeño de las consultas y la correlación entre el valor promedio del *entropy impurity* y el *R-Precision* obtenido. Estos comportamientos se pueden atribuir a aspectos independientes de la base de datos, como el efecto de saturación al utilizar muchos objetos por clase en el conjunto de entrenamiento. Los aspectos variables, como los valores distintos de *R-Precision* a nivel general, podrían ser atribuidos a las propiedades específicas de cada una, como la calidad de los descriptores, la compatibilidad entre ellos al ser combinados o a la construcción de las clases. En el caso de las imágenes, las clases tienen un alto grado de impureza debido a la naturaleza de las imágenes de representar más de un concepto al mismo tiempo. Probablemente al escoger imágenes más puras por cada clase el desempeño general aumentaría considerablemente.

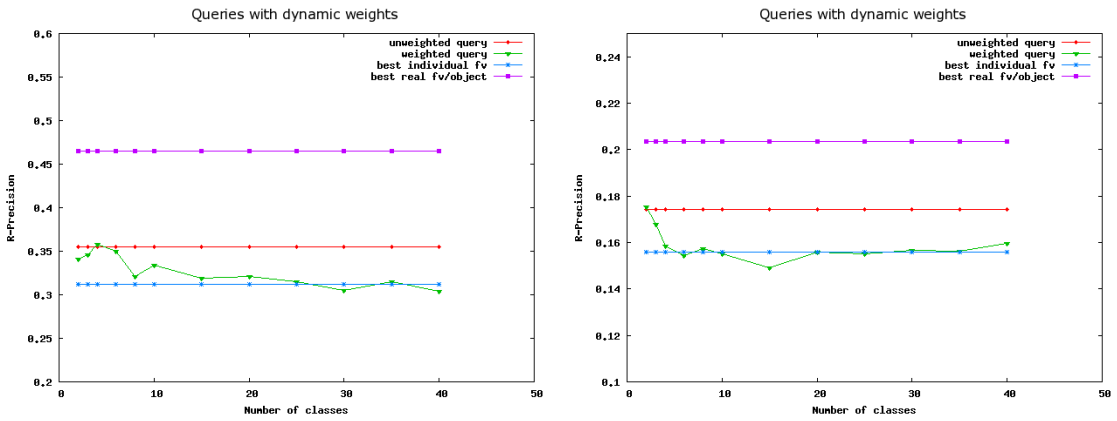

Figura 5.12: Comparación del efecto del número de objetos por clase en dos bases de datos distintas (Izquierda: PSB, Derecha: Imágenes). En ambos gráficos se utilizó un tamaño mínimo de 7 objetos, conservando 3 objetos por clase en el conjunto de entrenamiento. El comportamiento es prácticamente el mismo en las dos bases de datos.

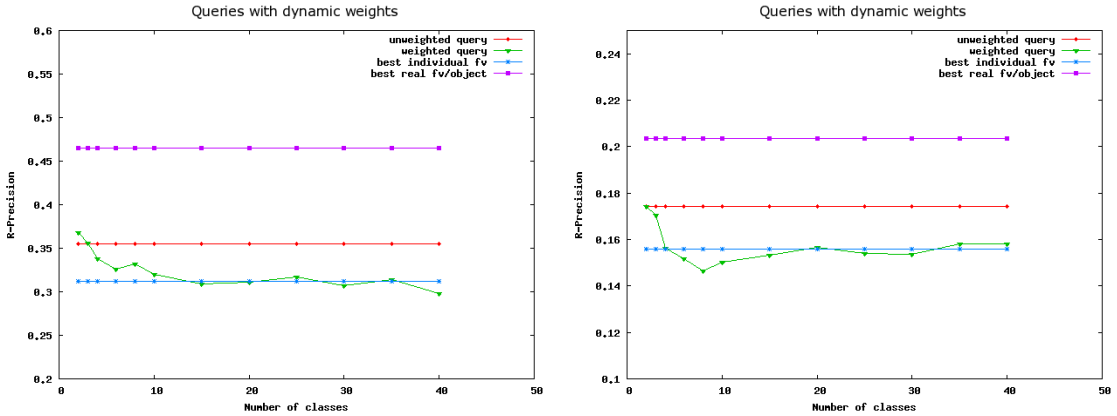

Figura 5.13: Comparación del efecto del número de clases en dos bases de datos distintas (Izquierda: PSB, Derecha: Imágenes). La tendencia a disminuir el valor del *R-Precision* promedio se observa en ambas bases de datos. Se utilizó un tamaño mínimo de 7 objetos, conservando 3 de ellos para el conjunto de entrenamiento.

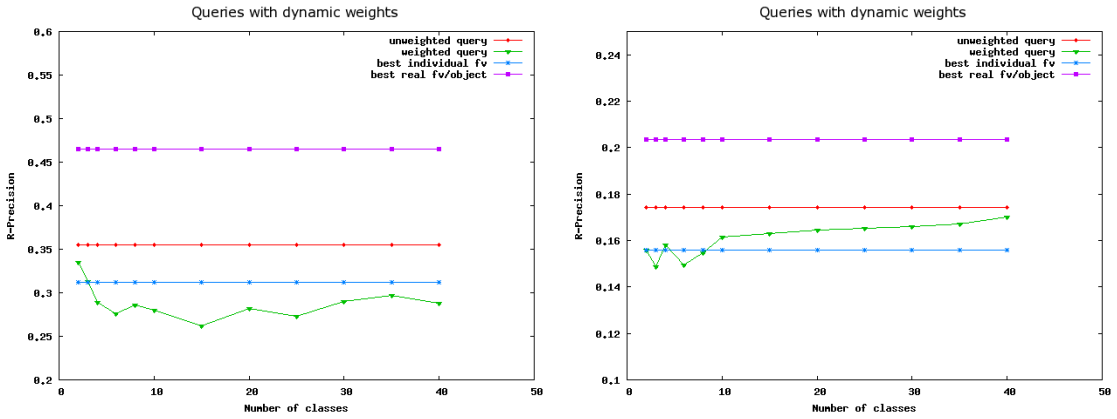

Figura 5.14: Comparación del efecto del número de clases en dos bases de datos distintas (Izquierda: PSB, Derecha: Imágenes). A diferencia de la PSB, en la base de datos de imágenes el *R-Precision* promedio mejora si se aumenta la cantidad de clases. Este caso se produce manteniendo 2 objetos por clase en el conjunto de entrenamiento.

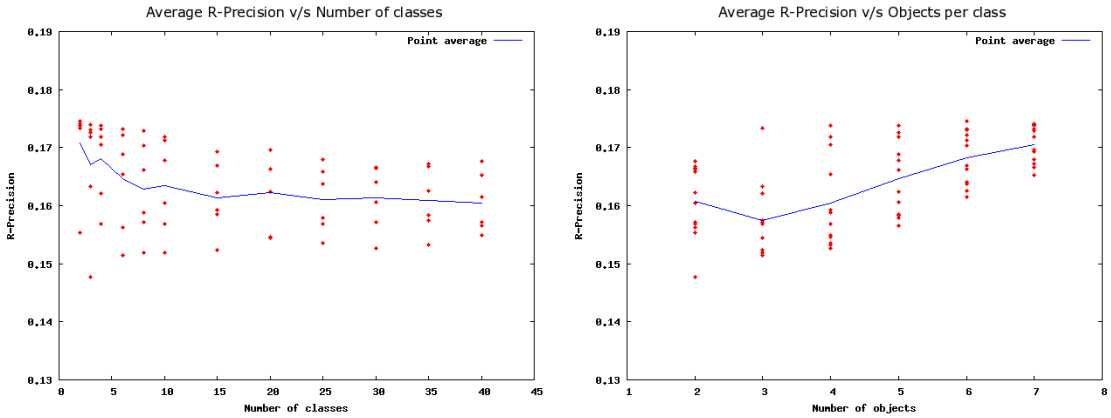

Figura 5.15: Visualización de los valores de *R-Precision* , agrupados según el número de clases y de objetos por clase.

Capítulo 6

Conclusiones

De los experimentos realizados en la sección anterior, junto con el análisis numérico y visual de sus resultados, es posible rescatar aspectos nuevos descubiertos en el área, comportamientos similares en distintas bases de datos, y posibles mejoras a los resultados finales alcanzados. A continuación se mencionan las conclusiones más importantes deducidas en cada aspecto.

En el caso de nuestro estudio principal, es notable destacar que el tamaño aceptable para un conjunto de entrenamiento es mucho menor que el de la MMDB original. Como las dos bases de datos utilizadas fueron casi del mismo tamaño, no fue posible deducir si este tamaño depende del tamaño original de la MMDB. Sin embargo, el efecto de saturación observado al utilizar muchas clases u objetos por clase sugiere que este tamaño debería permanecer en un valor bajo aún en bases de datos más grandes. Esto es un resultado sorprendente, ya que una consulta en una base de datos mucho más grande puede tomar un tiempo considerable. Sin embargo, con un conjunto muy pequeño de objetos se podría mejorar el desempeño de la búsqueda mediante una consulta extra, cuyo costo es prácticamente despreciable en relación a la consulta real.

Sobre los valores obtenidos para el indicador *R-Precision*, el más utilizado para evaluar cada experimento, es necesario destacar que las mejoras obtenidas por sobre los casos estáticos sólo fueron superiores al mejor vector individual y a la combinación de todos los vectores existentes. La mejor combinación de vectores es un caso bastante superior a los anteriores, dado que en general es calculado mediante fuerza bruta, evaluando todas las combinaciones posibles de vectores. En el caso que no se tenga esta información, o en que existan demasiados vectores para realizar las combinaciones, una aproximación de este valor utilizando un conjunto de entrenamiento pequeño podría lograr un desempeño intermedio entre la combinación estática de

vectores y la mejor combinación existente.

El otro aspecto confirmado a lo largo de todos los experimentos fue el grado de robustez de los resultados. Todos los experimentos que involucraban un grado de aleatoriedad fueron ejecutados repetidas veces para comprobar el grado de variación entre distintas ejecuciones. Los mismos resultados generales fueron observados a lo largo de cada ejecución. El único caso en que esto no ocurrió fue en el rango óptimo de tamaños para el conjunto de entrenamiento, cuando las clases eran escogidas aleatoriamente. Como fue mencionado anteriormente, las clases deberían ser escogidas cuidadosamente, para maximizar la eficacia en la asignación de pesos a cada distancia.

También hay que destacar la invariancia de los experimentos utilizando bases de datos distintas, lo cual era uno de los objetivos específicos de la memoria. Los resultados más importantes que fueron observados en ambas bases de datos son la correspondencia entre la calidad de un vector y el valor del *entropy impurity* obtenido con el mismo, lo cual es indispensable para asignar correctamente mejores pesos a los vectores de mejor desempeño. Junto con esto, cabe destacar la similitud entre los gráficos respectivos de ambas bases de datos, al utilizar los mismos valores de las variables involucradas. Con esto se puede deducir que al cambiar el tamaño del conjunto de entrenamiento, el desempeño se verá afectado de manera similar en bases de datos distintas.

Un último resultado interesante de rescatar es la relación lineal descubierta entre el *entropy impurity* promedio para un vector y el *R-Precision* respectivo. Este resultado indica que la calidad de un FV se puede evaluar mediante la cantidad de objetos relevantes en la consulta, o mediante la coherencia de la misma, y que ambas evaluaciones entregan un resultado similar en promedio. La correlación entre estas variables fue muy alta, y esto también fue observado en ambas bases de datos.

Como parte de las mejoras posibles a este trabajo se puede mencionar el cálculo de los pesos para escoger un subconjunto de vectores a combinar. A lo largo de este trabajo, los experimentos se realizaron combinando todos los vectores con sus respectivos pesos. Sin embargo, utilizar todos los vectores podría no ser la mejor opción, como fue mostrado en la Tabla 4.6, donde el mayor *R-Precision* posible se obtiene con un número menor de vectores a combinar. Teniendo esto en cuenta, sería interesante estudiar cómo varía el desempeño si los pesos son

utilizados para seleccionar los mejores vectores para una consulta (los cuales serían los vectores con los pesos más altos), y una vez seleccionados éstos, comprobar si existe una diferencia entre combinarlos con pesos unitarios o con los mismos pesos calculados mediante *entropy impurity*.

Otra opción a futuro para mejorar la construcción del conjunto de entrenamiento es investigar la mejor forma de escoger las clases a mantener. Dado que esto no formaba parte de nuestro estudio, se hizo una verificación sencilla de las combinaciones que logran un mayor desempeño, obteniendo un número reducido de clases dentro de las mejores combinaciones. Las mejores clases, por lo tanto, no deberían ser escogidas aleatoriamente, aunque la manera de encontrarlas sin utilizar fuerza bruta aún no es clara. Lo que probablemente sea cierto es que clases muy cercanas funcionarán mal en el conjunto de entrenamiento, ya que éstas distorsionan de manera innecesaria el verdadero rendimiento que se puede obtener.

Como comentario final, sería interesante utilizar una implementación de este conjunto en alguna base de datos real. En este caso se podría determinar con mayor certeza la mejora en la calidad de una consulta, cuando se utiliza una cantidad extra muy pequeña de recursos. De esta forma sería posible evaluar en una situación real si el tiempo extra dedicado a calcular esta información es compensado con una mejor calidad de la respuesta que es entregada finalmente por la base de datos.

Bibliografía

[1] B. Bustos, D. Keim, D. Saupe, T. Schreck, D. Vranić: *An experimental effectiveness comparison of methods for 3D similarity search*. International Journal on Digital Libraries (2006)

[2] B. Bustos. *Index structures for similarity search in multimedia databases*. Ph.D. thesis, Department of Computer and Information Science, University of Konstanz. June, 2006

[3] B. Bustos, D. Keim, D. Saupe, T. Schreck, D. Vranić: *Using entropy impurity for improved 3D object similarity search*. IEEE International Conference on Multimedia and Expo, ICME 2004. Páginas 1303-1306

[4] R. Baeza-Yates, B. Ribeiro-Neto: *Modern Information Retrieval*. Páginas 73-97. Editorial Addison-Wesley-Longman. Edición 2004.

[5] *Princeton Shape Benchmark* (documentación, base de datos y utilidades): `http://shape.cs.princeton.edu/benchmark/`. Fecha de acceso: 15-03-2008

[6] *Vectores Característicos para Princeton Shape Benchmark*:

`http://www.gris.informatik.tu-darmstadt.de/~tschreck/code/fv/`.
Fecha de acceso: 21-03-2008

[7] *Especificación formato OFF para modelos 3D*:

`http://local.wasp.uwa.edu.au/~pbourke/dataformats/oogl/#OFF`.
Fecha de acceso: 30-06-2008

[8] *Especificación formato de clasificación de objetos*:

`http://shape.cs.princeton.edu/benchmark/documentation/-classification_format.html`. Fecha de acceso: 30-06-2008