

Recuperación de imágenes basada en texto y contenido visual mediante redes neuronales

Diego Adrián Castro

Directora
Leticia María Seijas
[lseijas@dc.uba.ar](mailto:lseijs@dc.uba.ar)



Tesis de Licenciatura
Ciencias de la Computación
Universidad de Buenos Aires
Facultad de Ciencias Exactas y Naturales
Departamento de Computación
Octubre de 2009

A la memoria de mi abuela Nina

Resumen

Durante los últimos años, la cantidad de información visual que se produce a diario viene experimentando un fuerte crecimiento. Desde imágenes generadas por satélites, cámaras de vigilancia y hasta fotos obtenidas por cámaras digitales contribuyen a formar una masa de información que de a poco se vuelve inmanejable. A lo largo del tiempo, se han descripto las imágenes a través de texto (TBIR) y de sus características visuales (CBIR), pero estos enfoques aún presentan desventajas. En los últimos años se ha comenzado a estudiar sistemas híbridos que combinan ambas variantes buscando potenciar las virtudes de cada una. A diferencia de TBIR, CBIR es un área mucho menos madura y presenta numerosos desafíos, entre ellos la definición de descriptores e índices adecuados. En este trabajo utilizamos histogramas de color para describir a las imágenes y analizamos el uso de mapas auto-organizados (SOM) como índice, ya que permiten trabajar con descriptores de alta dimensionalidad (caso típico en CBIR). Proponemos una función de *scoring* para imágenes con la idea de descartar imágenes irrelevantes en los resultados y también una variante para los SOM que mejora los tiempos de entrenamiento y recuperación (ParBSOM). Además, aplicamos dichas técnicas en el contexto de un sistema híbrido y proveemos resultados empíricos para evaluar su desempeño. Por último, presentamos un software propio denominado *Envision* que aplica los métodos estudiados.

Abstract

In the last few years there has been a dramatic increase in the visual information available. Images generated from satellites, surveillance cameras and even digital cameras produce a huge amount of information that gradually becomes more difficult to handle. Typically, images are described by their textual content (TBIR) or by their visual features (CBIR). However, these approaches still show many problems. Recently, it was introduced the hybrid approach which combines both characteristics to improve the benefits of using text and visual content separately. CBIR nowadays is still far from being as well-matured as TBIR and presents many challenges such as defining suitable descriptors and index structures. In this work we use color histograms to describe images and study how Self-Organizing Maps (SOM) can be used as an index in CBIR. SOM are an interesting alternative as they allow us to work with high-dimensional descriptors (typical case in CBIR). We propose a scoring function for images which eliminates irrelevant images from the results list and we also introduce a new SOM model that improves training and retrieval times (ParBSOM). In addition, we study how these techniques can be applied to the hybrid approach and provide computational results to assess their performance. Finally, we develop a system known as *Envision*, which implements all the studied methods.

Agradecimientos

Sin lugar a dudas, a la hora de hablar de agradecimientos lo primero que se me viene a la mente son mis papás. El hecho de que esté presentando mi tesis y esté a punto de recibirme es un logro que estoy seguro que ellos también sienten como propio. Siempre me apoyaron en todo, confiaron en mí y me aconsejaron para que pudiera lograr mi objetivo de recibirme. ¡¡¡Gracias Papus!!!

Este camino arrancó allá por el 2002 cuando empecé el CBC y, no voy a mentir, fue bastante duro. Gran parte de estos últimos años me los pasé en mi querido Pabellón 1 (y sí, después de estar tanto tiempo ahí, uno lo empieza a querer). No me voy a olvidar de las horas frente a la compu haciendo TP en los labos, de la infinidad de cafés de la Noriega que tomé, del tiempo que me pasé estudiando en esas mesas del bar del Pabellón 1 (sobre todo en las mesas de atrás, lugar conocido como “Zona liberada” y reservado para momentos donde había que intensificar el estudio y tener máxima concentración). Y ni hablar de las eternas horas de cursada (aprovecho para confesar que siempre me costó mantener la concentración en las clases). ¡¡Y tampoco me voy a olvidar de los 24 y 25 de diciembre que pasé estudiando para aprovechar la última fecha de final del año!! Bueno, hubo que darle mucha dedicación y resignar muchas cosas, pero ahora que estoy llegando al objetivo puedo decir que siento que valió la pena.

Abreviaturas

CBIR	Content Based Image Retrieval (Recuperación basada en contenido)
TBIR	Text Based Image Retrieval (Recuperación basada en texto)
VIR	Visual Information Retrieval (Recuperación de información visual)
HSV	Espacio de color Hue-Saturation-Value
RGB	Espacio de color Red-Green-Blue
SOM	Self-Organizing Maps (Mapas auto-organizados)
BSOM	Batch SOM
ParSOM	Parallel SOM
ParBSOM	Parallel Batch SOM
BMU	Best Matching Unit (Neurona ganadora)

Índice general

1. Introducción	1
1.1. Objetivos	2
1.2. Trabajo Relacionado	3
1.3. Estructura de la Tesis	4
2. VIR y su teoría	5
2.1. Qué es VIR	5
2.2. Etapas para construir un sistema de VIR	6
2.3. CBIR	6
2.3.1. Extracción de características	6
2.3.2. Indexación	10
2.4. TBIR	11
2.4.1. Extracción de características	11
2.4.2. Indexación	12
2.5. Recuperación	13
2.6. Evaluación	14
2.6.1. Bases de datos de imágenes	15
2.6.2. Métricas	16
2.7. Aplicaciones	19
3. Histograma de Color	21
3.1. Descripción general	21
3.2. ¿Por qué utilizar Histogramas de Color en CBIR?	22
3.3. Espacios de Color	22
3.3.1. RGB	22
3.3.2. HSV	23
3.4. Cuantización de los colores	24
3.5. Cantidad de <i>bins</i> del histograma	24
3.6. Función de distancia entre histogramas	24
3.7. Función de <i>scoring</i> para histogramas	24
3.8. Construcción del histograma	25
3.9. Experimentación	29
3.9.1. Descripción	29
3.9.2. Resultados	29

4. Self-Organizing Maps (SOM)	32
4.1. Introducción	32
4.2. Arquitectura	33
4.2.1. Construcción de un SOM	33
4.3. Características principales del SOM	36
4.4. Variantes del SOM	37
4.4.1. BSOM	37
4.4.2. ParSOM	38
4.4.3. Variante propuesta: ParBSOM	38
4.4.4. Otras variantes	40
4.5. Experimentación	42
4.5.1. Descripción	42
4.5.2. Resultados	43
5. Indexación CBIR	47
5.1. Introducción	47
5.2. Construcción del índice	48
5.3. Recuperación a través del índice	48
5.4. Experimentación	50
5.4.1. Descripción	50
5.4.2. Resultados	50
6. Recuperación Híbrida	53
6.1. Introducción	53
6.2. Estructura de un Sistema Híbrido	53
6.3. Estrategias de combinación de resultados de TBIR y CBIR	55
6.3.1. Votación	55
6.3.2. Refinamiento	55
6.4. Sistema propuesto: <i>Envision</i>	56
6.5. Experimentación	56
6.5.1. Descripción	56
6.5.2. Resultados	59
7. Conclusiones y trabajos futuros	61
7.1. Conclusiones	61
7.2. Trabajos futuros	63
A. Demostraciones	64
A.1. Normalización de norma L1	64
A.2. Complejidad temporal de SOM y BSOM	65
Bibliografía	70

Índice de figuras

2.1. Semantic Gap	7
2.2. Ambigüedad: ¿Una mujer joven o una anciana?	7
2.3. Contexto: El cuadrado A y B tienen el mismo tono de gris, pero el contexto nos hace pensar lo contrario	8
2.4. Conocimientos previos/Factores culturales: ¿Un hombre con bigotes dado vuelta? Si rotamos la imagen, se observa que no es así...	8
2.5. Problemas lenguaje natural: Imprecisión	11
2.6. Problemas lenguaje natural: Ambigüedad	12
2.7. Índice invertido (http://developer.apple.com)	13
2.8. El mismo edificio desde diferentes perspectivas (imágenes de la base ZuBuD) . .	15
2.9. Imágenes de la base UCID	16
2.10. Imágenes de la base UK Bench	17
2.11. Imagen de la ImageCLEF 2007 junto a su texto asociado	17
3.1. Espacio de color RGB	23
3.2. Espacio de color HSV	23
3.3. Ejemplo de uso de umbral para limitar la cantidad de resultados	26
3.4. Gráfico de F-Measure vs. Umbral para base de datos ZuBuD	30
3.5. Gráfico de F-Measure vs. Umbral para base de datos UCID	30
3.6. Gráfico de F-Measure vs. Umbral para base de datos UK Bench	31
4.1. Estructura típica de un SOM	34
4.2. Tasa de aprendizaje lineal	35
4.3. Función de vecindad Gaussiana	36
4.4. En ParSOM, la red es dividida en varias regiones y cada nodo de procesamiento trabaja exclusivamente con ella [1]	39
4.5. Estructura de un TS-SOM de 3 niveles [2]	41
4.6. GH-SOM de 4 niveles [3]	42
4.7. Tiempos para 1.000 muestras de dimensión 250	44
4.8. Tiempos para 1.000 muestras de dimensión 500	44
4.9. Tiempos para 5.000 muestras de dimensión 250	45
4.10. Tiempos para 5.000 muestras de dimensión 500	45
4.11. Tiempos para 10.000 muestras de dimensión 250	46
4.12. Tiempos para 10.000 muestras de dimensión 500	46
5.1. Recuperación de imágenes a través de un SOM	49
5.2. Gráfico de F-Measure para las bases de datos y las técnicas de indexación . .	51

ÍNDICE DE FIGURAS

5.3. Tiempo requerido para recuperar una imagen para Fuerza Bruta y para un SOM en las bases de datos	52
6.1. Estructura de un sistema de VIR siguiendo el enfoque híbrido	54
6.2. <i>Envision</i> : La consulta puede consistir en texto, una imagen o ambas alternativas	57
6.3. <i>Envision</i> : A partir de texto y una imagen, se recuperan imágenes relevantes para la consulta	58
6.4. TBIR vs. TBIR+CBIR para diferentes métricas en la base ImageCLEFphoto 2007	59

Índice de tablas

2.1.	Información detallada de las BDs de imágenes	15
2.2.	Variables utilizadas para definir las métricas	18
3.1.	RGB 8x8x8	29
3.2.	HSV 32x4x4	29
4.1.	Parámetros usados para comparar tiempos de variantes de SOM	43
4.2.	Comparación de tiempos para variantes del SOM (10 épocas de entrenamiento)	43
5.1.	Comparación de la calidad de recuperación de las técnicas de indexado	50
5.2.	Pérdida con respecto a F-Measure de las técnicas de indexado	51
5.3.	Comparación del tiempo requerido por las técnicas de indexado para recuperar una imagen	51
6.1.	Comparación de tipos de recuperación en la base ImageCLEFphoto 2007	59

Capítulo 1

Introducción

Durante los últimos años, la cantidad de información visual que se produce a diario viene experimentando un fuerte crecimiento. Desde imágenes generadas por satélites, cámaras de vigilancia y hasta fotos obtenidas por cámaras digitales contribuyen a formar una masa de información que de a poco se vuelve inmanejable. Actualmente, existe una creciente demanda de métodos de búsqueda que sean capaces de recuperar imágenes en grandes bases de datos, sobre todo en campos y actividades tan diversas como la Web, las galerías de arte y museos, investigación criminal, identificación de personas, identificación de logotipos, diseño industrial, diagnóstico médico, entre otras. Sin dudas, este nuevo escenario trae aparejados problemas para acceder a esta gran cantidad de información y es por este motivo que la recuperación de imágenes asoma como una de las áreas de investigación que tendrá mayor preponderancia en los próximos años.

El área de recuperación de imágenes, conocida también como *Visual Information Retrieval* (VIR), desde tiempo atrás ha comenzado a estudiar diferentes técnicas para realizar búsquedas eficientes en bases de datos de imágenes. Existen tres enfoques: recuperación basada en texto, basada en contenido visual y una combinación de ambas características.

En la recuperación basada en texto o *Text Based Image Retrieval* (TBIR) [4], las imágenes son descriptas a través de un conjunto de palabras. A su vez, para buscar imágenes similares se utiliza una consulta basada exclusivamente en texto. Si bien este enfoque es ampliamente utilizado en la actualidad, presenta algunas desventajas. Los textos que describen a las imágenes son claramente subjetivos y pueden ser vagos e imprecisos, dificultando las tareas de recuperación basadas en texto.

Intentando evitar los problemas inherentes a TBIR, surge otro enfoque: recuperación basada en contenido o *Content Based Image Retrieval* (CBIR) [4]. En lugar de utilizar un rotulado manual de la imagen, la idea es usar información acerca de su contenido visual como puede ser el color, la forma o la textura. Al mismo tiempo, para la recuperación se utilizará como consulta directamente a una imagen.

Así como la recuperación basada en texto presenta desventajas, la basada en contenido no es la excepción. Describir a las imágenes con características de bajo nivel como el color o la forma puede provocar que imágenes conceptualmente diferentes sean consideradas similares

(problema conocido como *semantic gap*). Por este motivo, en los últimos años se ha comenzado a estudiar la recuperación híbrida, combinando la descripción textual y visual con el fin de reducir los defectos de cada enfoque por separado y, al mismo tiempo, potenciar sus virtudes.

A diferencia de TBIR donde es más sencilla y eficiente la construcción y utilización de estructuras de índices, CBIR aún presenta desafíos con respecto a los métodos más convenientes para realizar las búsquedas. En primer lugar, es necesario definir el conjunto de características visuales que se extraerán de la imagen. Luego, la estructura que tendrá el índice, por ejemplo redes neuronales [5], R-trees [6], KD-trees [6], entre otras. Estas decisiones son fundamentales ya que tanto el descriptor como el índice determinarán la calidad y eficiencia de la recuperación en sistemas de CBIR.

Una de las técnicas de recuperación que ha acaparado gran atención es la de los *mapas auto-organizados de Kohonen*, también conocidos como *Self-Organizing Maps* (SOM) [7, 8, 5]. Los SOM son uno de los principales representantes del paradigma del aprendizaje no supervisado. Actúan como clasificadores de imágenes, asociando un conjunto de imágenes a cada neurona. Generan una estructura de salida que refleja el orden topológico de las imágenes de entrada en función del descriptor definido. La formación de agrupamientos de neuronas hace que imágenes similares se encuentren cercanas, facilitando las tareas de recuperación.

Si bien los SOM aparecen como una alternativa interesante para usar en CBIR, existen puntos sobre los cuales aún falta avanzar. Para grandes bases de datos, en los SOM tradicionales la etapa de entrenamiento se vuelve muy compleja y costosa. Algunas variantes como *BSOM* [8], *ParSOM* [9], *TS-SOM* [10] o *GH-SOM* [3] intentan atacar estos aspectos y hoy en día nuevos tipos de SOM se encuentran bajo desarrollo.

Actualmente, se han desarrollado y se está trabajando en numerosos sistemas de VIR con diferentes características. La existencia de bases de datos de imágenes como la *ZuBuD* [11], *UCID* [12], *UK Bench* [13] e *ImageCLEFphoto 2007* [14], así como también la organización de eventos como *Benchathlon*¹ e *ImageCLEF*² brindan la posibilidad de comparar sistemas de VIR, contribuyendo fuertemente a generar avances en el área.

1.1. Objetivos

Los principales objetivos de este trabajo son:

- Presentar una estrategia híbrida para la recuperación de imágenes basada en contenido visual y textual, junto con el desarrollo de un software denominado *Envision*, que permite aplicar dicha técnica así como también TBIR y CBIR.
- Describir el estado del arte y los fundamentos teóricos de VIR necesarios para el desarrollo de este trabajo.
- Presentar resultados experimentales de los métodos propuestos usando bases de datos de imágenes de acceso libre, utilizadas en trabajos del área o competencias internacionales.

¹<http://www.benchathlon.net/>

²<http://imageclef.org/>

- Presentar una función de scoring para imágenes descriptas a través de histogramas de color que permite eliminar imágenes irrelevantes para una consulta dada.
- Evaluar la capacidad de los SOM a la hora de indexar bases de datos de imágenes.
- Proponer una variante a los SOM (denominada ParBSOM), que permite entrenar a las redes y recuperar imágenes en forma más eficiente.

1.2. Trabajo Relacionado

Los orígenes del área de VIR se remontan hacia fines de los años 70' y desde ese momento ha sido un área de investigación muy activa. Los primeros trabajos pertenecieron a la comunidad de las bases de datos relacionales y se enfocaban en la recuperación basada en texto (TBIR).

A principios de los años 90', aparece por primera vez mencionada la técnica de recuperación basada en contenido (CBIR). En un trabajo de T. Kato [15], el término CBIR fue utilizado para describir a un sistema que recuperaba imágenes de una base de datos basándose en el color y la forma.

El surgimiento de esta nueva área vino acompañado con la aparición de numerosos sistemas de recuperación [16]. Uno de los pioneros fue IBM, quien desarrolló el sistema QBIC [17] en el año 1995. Este producto usaba descriptores basados en color, textura y forma. Con el paso de los años, fueron apareciendo más sistemas con diferentes variantes, entre los cuales se destacan Photobook [18] (realizado en el MIT), Blobworld [19] (desarrollado en UC Berkeley), SIMBA [20], FIRE [21], entre otros.

Con el tiempo, varios de estos sistemas fueron avanzando sobre otras áreas. Aparecieron buscadores de imágenes en Internet como Webseer [22] en el año 1996 y Webseek [23] en el año 1997. A su vez, en 1999 motores populares de bases de datos relacionales como IBM DB2 y Oracle incluyeron herramientas a sus productos para facilitar la recuperación de imágenes por contenido visual, acercando el área de CBIR al ámbito de la industria. Actualmente se sigue trabajando sobre CBIR, buscando desarrollar sistemas eficientes y que se acerquen a la percepción de similitud del humano.

La constante aparición de nuevos sistemas, cada uno con diferentes descriptores visuales, motivó a la gente del *MPEG* (Moving Picture Experts Group) a buscar desarrollar un estándar para la descripción de imágenes. Se definieron una serie de descriptores que se publicaron en el año 2001, sin embargo al día de la fecha no han tenido gran repercusión en el área de CBIR. En la actualidad, conviven gran cantidad de descriptores y uno de los más simples y populares es el histograma de color [24].

A diferencia de la recuperación de texto (IR, Information Retrieval [25]) donde existen eventos masivos como *TREC* (Text REtrieval Conference³) donde se evalúan y comparan sistemas con diferentes bases de datos, en VIR todavía no se ha logrado organizar eventos de tal magnitud e importancia. El Benchathlon se realizó por primera vez en 2001 y su objetivo

³<http://trec.nist.gov/>

fue crear un marco de discusión y comparación de diferentes técnicas de CBIR. Este evento se realizó hasta 2004 y no logró afirmarse como se esperaba inicialmente. Desde el año 2003 hasta el día de la fecha, anualmente viene desarrollándose la ImageCLEF, que abarca técnicas basadas en texto, en contenido visual e híbridas.

Con respecto a sistemas de CBIR basados en SOM, uno de los primeros fue desarrollado por la Universidad de Helsinki en el año 1999. Se publicó bajo el nombre de *PicSOM* [2] y actualmente se continúa avanzando sobre diferentes aspectos del mismo. *FACERET* [26] fue creado por la Universidad de Santiago de Chile en el año 2002 y se utiliza para buscar en forma interactiva rostros humanos utilizando SOM. Uno de los más recientes es el *GalsOM* [27], sistema desarrollado en República Checa, publicado en el año 2007 y que se aplica a la detección de robo de imágenes de bases de datos comerciales.

1.3. Estructura de la Tesis

En el Capítulo 2 se realiza una introducción a la teoría relacionada a VIR, analizando los enfoques basados en texto, en contenido visual e híbridos. Además, se presentan las bases de datos de imágenes y las métricas que a lo largo del trabajo se utilizan para comparar sistemas de VIR.

Una vez presentada la estructura general de sistemas de recuperación de información visual, en el Capítulo 3 se estudia una de las estructuras más utilizadas para describir imágenes: los histogramas de color. Se analizan diferentes variantes con respecto al espacio de color utilizado y se presenta una función de *scoring* que permite eliminar imágenes irrelevantes en forma sencilla.

Luego, en el Capítulo 4 se introduce el tema de las redes neuronales, más precisamente el de los *Self-Organizing Maps* (SOM). Allí se presentan diferentes variantes para el SOM tradicional y se propone la utilización de un SOM que combina paralelismo con entrenamiento *batch*: el ParBSOM.

En el Capítulo 5 se describe cómo es posible utilizar a los SOM para indexar imágenes en sistemas de CBIR. Se realiza una comparación contra el algoritmo exacto de *Fuerza Bruta* (búsqueda lineal de las imágenes), midiendo tanto la calidad de recuperación como también el tiempo de respuesta.

Una vez presentados los temas anteriores, durante el Capítulo 6 se realiza un estudio sobre los sistemas híbridos, que combinan la recuperación basada en texto y en contenido visual. En base a lo investigado en los capítulos previos, se introduce un sistema híbrido desarrollado por nosotros bajo el nombre de *Envision*. Sobre el final, se presentan resultados que mejoran los obtenidos por enfoques basados exclusivamente en descripciones textuales o visuales.

Por último, en el Capítulo 7 se incluyen las conclusiones generales, los aportes realizados y los posibles trabajos a futuro.

Capítulo 2

VIR y su teoría

Los orígenes del área de *Visual Information Retrieval* (VIR) se remontan a fines de la década del 70' y desde ese momento ha sido una de las áreas de investigación más activas. En este capítulo detallaremos en qué consiste VIR y hablaremos en forma detallada de su teoría relacionada.

2.1. Qué es VIR

En los últimos años, el área de recuperación de imágenes ha adquirido gran importancia debido a la enorme cantidad de información visual que se genera diariamente. VIR es considerada un área relativamente nueva y por ello todavía se presentan una gran cantidad de desafíos y variantes.

Existen tres enfoques para la recuperación de imágenes:

- **Recuperación basada en texto** (TBIR): Las imágenes son descriptas a través de texto.
- **Recuperación basada en contenido** (CBIR): Las imágenes son descriptas a través de características visuales (color, textura, forma, entre otras).
- **Recuperación basada en texto y contenido** (enfoque híbrido): Las imágenes son descriptas a través de texto y de sus características visuales.

TBIR es la variante que más antigüedad tiene y al mismo tiempo es la más utilizada en la actualidad. Un claro ejemplo de esto es que motores de búsqueda como *Google* y *Yahoo!* usan este enfoque.

El área de CBIR surgió tiempo después como alternativa a la recuperación por texto. Si bien en los últimos años se ha incrementado considerablemente la cantidad de trabajos sobre CBIR, aún presenta numerosos desafíos.

Tanto la recuperación basada en texto como la basada en contenido poseen sus desventajas. En cuanto a TBIR, los textos que describen a las imágenes son claramente subjetivos y pueden ser vagos e imprecisos. A su vez, como CBIR utiliza características de bajo nivel de la imagen, es posible que se consideren parecidas a imágenes semánticamente diferentes. Con

la idea de reducir las desventajas y potenciar las virtudes de cada enfoque, surgió la idea de utilizar sistemas híbridos. En los últimos años se comenzaron a investigar diferentes técnicas para combinar la descripción a través de texto y de las características visuales. Durante este capítulo trataremos cada uno de estos temas en detalle.

2.2. Etapas para construir un sistema de VIR

La construcción de un sistema de VIR no es una tarea sencilla, dado que se busca conseguir una recuperación que satisfaga las necesidades del usuario tanto en calidad de los resultados como también en tiempo de respuesta. La forma de describir a las imágenes es una parte crucial en el sistema, ya que debe intentar acercarse lo más posible a la percepción de similitud de los usuarios. Al mismo tiempo, la generación de estructuras de índices definirá en gran medida la eficiencia del sistema y también impactará fuertemente en la calidad de los resultados generados.

En las siguientes secciones profundizaremos sobre las diferentes etapas que se llevan a cabo en un sistema de VIR, pero antes, a modo de resumen, las presentaremos brevemente:

1. **Extracción de características:** A cada imagen se le extrae información necesaria para describirla.
2. **Indexación:** La información extraída a cada imagen es almacenada en una estructura de índice para una eficiente recuperación.
3. **Recuperación:** A partir de una imagen, se buscan las imágenes más parecidas a través del índice.
4. **Evaluación:** Se mide la calidad de recuperación del sistema.

2.3. CBIR

2.3.1. Extracción de características

A la hora de construir un sistema de CBIR, la definición de un descriptor de imágenes adecuado es fundamental. En la actualidad existe un gran número de descriptores y todavía no se ha alcanzado un consenso acerca de cuál es el más adecuado. La mayor dificultad que se presenta es encontrar un descriptor que decida si imágenes son similares de la forma en la que los humanos lo harían. La brecha que existe entre la caracterización que puede realizar una computadora y la de un humano se conoce en la literatura como *semantic gap* (Fig. 2.1).

El denominado *semantic gap* se da por cuestiones inherentes al comportamiento humano. A la hora de interpretar una imagen deben tenerse en cuenta factores como por ejemplo [28]:

- *Ambigüedad:* Una imagen puede interpretarse de diferentes formas (Fig. 2.2).
- *Contexto:* El cerebro humano es sensible al contexto (Fig. 2.3).



Figura 2.1: Semantic Gap



Figura 2.2: Ambigüedad: ¿Una mujer joven o una anciana?

- *Conocimientos previos/Factores culturales:* La información previa y las influencias culturales puede llevar a diferentes interpretaciones (Fig. 2.4).

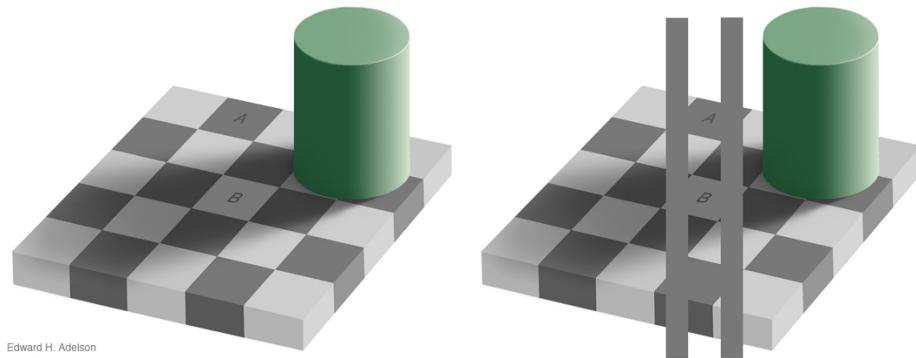
Sin lugar a dudas, las cuestiones anteriores difícilmente puedan ser emuladas con gran exactitud por una computadora, ya que éstas ven a las imágenes como un conjunto de píxeles. Los descriptores visuales existentes se basan en un análisis de los píxeles que conforman la imagen y suelen describir a las imágenes a través del color, de la textura y/o de las formas.

Más allá de intentar aproximar la percepción de los humanos, deben tenerse en cuenta otras cuestiones a la hora de seleccionar un descriptor adecuado. Tanto el *costo computacional* como el *costo espacial* son temas de gran interés.

A continuación presentaremos diferentes clases de descriptores. Vale aclarar que lo que sigue no es un listado exhaustivo sino una selección de los más representativos y populares en la actualidad.

Color

El color es una de las características más simples y sencillas para describir el contenido de una imagen. Su uso es muy popular dentro de la comunidad de CBIR ya que es fácil de entender e interpretar y el costo computacional y espacial suele ser bajo. La elección del espacio de



Edward H. Adelson

Figura 2.3: Contexto: El cuadrado A y B tienen el mismo tono de gris, pero el contexto nos hace pensar lo contrario



Figura 2.4: Conocimientos previos/Factores culturales: ¿Un hombre con bigotes dado vuelta?
Si rotamos la imagen, se observa que no es así...

color así como también la cuantización de los colores son fundamentales a la hora de utilizar un descriptor de este tipo.

El descriptor más representativo de esta clase es el *histograma de color*, presentado por Swain y Ballard [24]. En el histograma de color, el espacio de color es dividido en N colores y se cuenta la cantidad de píxeles de cada color, obteniendo una representación de las frecuencias de cada color en la imagen.

Existen otros descriptores que trabajan con color, como por ejemplo *color layout* [16] que representa la distribución espacial de los colores, *color moments* [16] que utiliza los momentos de color en lugar de la cuantización, entre otros.

Textura

La textura se define como las propiedades visuales que tiene una superficie. Contiene información importante acerca de la estructura de las superficies y su relación con el ambiente que las rodea.

Se han desarrollado una gran cantidad de trabajos sobre textura en el área de *Pattern Recognition* y *Computer Vision*, que luego han sido aplicados a sistemas de CBIR. Entre los descriptores de textura más populares se encuentran los *wavelets* [29] y los descriptores desarrollados por *Tamura* y *Gabor* [16].

Forma

Una manera de describir una imagen es a través de las formas de los objetos que la componen. A diferencia de los descriptores visuales que utilizan color o textura, las formas no han sido tan populares dentro del área de CBIR. La detección de formas está íntimamente relacionada con la tarea de reconocimiento de objetos dentro de una imagen, tarea que suele requerir intervención humana dada su dificultad para automatizar el proceso.

Dentro de los descriptores de formas, se destacan los de *Fourier* y *Zernike* [16].

Otros

En muchos casos, los descriptores son diseñados pensando en un dominio específico. Uno de los ejemplos más claros es el del reconocimiento facial, donde se utiliza un descriptor conocido como *eigenfaces* [26] que sirve para describir caras.

MPEG-7, en la búsqueda de consenso

Debido a la proliferación de descriptores visuales, el *Moving Picture Experts Group* (MPEG) decidió comenzar a trabajar en una serie de descriptores que se convirtieran en estándar para el área de aplicaciones multimedia. Su idea estaba centrada en encontrar descriptores eficientes tanto en costo computacional como espacial.

En el año 2001, el grupo MPEG publicó una serie de descriptores conocidos como *MPEG-7 Features*. Además de definir los descriptores, se realizaron implementaciones de referencia para

trabajos futuros.

Hasta el día de la fecha este trabajo no ha tenido gran repercusión dentro del área y, si bien no se ha logrado cumplir el objetivo inicial de convertirse en un estándar en CBIR, fue una interesante y valiosa idea.

2.3.2. Indexación

Dentro de un sistema de CBIR, la forma de indexar los descriptores generados para cada imagen no es una tarea trivial. En su forma más sencilla, una base de datos de imágenes es simplemente una colección de imágenes. Sin embargo, la necesidad de encontrar estructuras eficientes que permitan un acceso rápido a la información se hace más notoria a medida que las bases de datos aumentan su tamaño. Los requerimientos computacionales a la hora de recuperar las imágenes en forma eficiente vuelven indispensable el uso de estructuras que permitan un rápido acceso. Cuando hablemos de *índices*, nos referiremos a estructuras adicionales sobre los datos originales que se utilizan para permitir una recuperación eficiente.

Un punto importante que debe tenerse en cuenta es que tanto la indexación como la extracción de características de las imágenes se realizan en forma *offline*, durante la construcción del sistema. Durante el funcionamiento *online* de la aplicación, sólo se procesan las consultas de los usuarios. Por este motivo, a la hora de elegir un índice se privilegia su eficiencia en la recuperación sobre el tiempo que demora en construirse. Cabe aclarar que en ambientes dinámicos, donde la base de datos puede cambiar con el tiempo, cada intervalos regulares o cuando se supera un número dado de cambios en la base, se suelen reconstruir los índices en *background* y en forma *offline* [30].

Existe una gran cantidad de opciones a la hora de elegir un tipo de índice para CBIR, sin embargo no existe un consenso acerca de cuál estructura es la más adecuada. Una característica que debe tenerse en cuenta a la hora de elegir el índice, es que los descriptores visuales suelen tener una alta dimensionalidad (aproximadamente, mayor a 100) [4].

El índice más sencillo consiste en almacenar los descriptores sin ningún orden en especial y luego realizar una búsqueda lineal sobre las imágenes (a lo largo del trabajo, a este enfoque lo llamaremos *Fuerza Bruta*). Sin embargo, esta variante puede resultar impráctico para grandes bases de datos.

Una alternativa son los índices con estructura de árbol como KD-trees [6], R-trees [6], entre otros. Si bien estas variantes proveen una búsqueda de orden logarítmico con respecto al tamaño de la base de imágenes, no escalan para dimensiones mayores a 20 [4]. Además, la mayoría asume que la distancia entre descriptores es la Euclídea, a pesar de que en muchos casos esta distancia no sea la más conveniente para simular la percepción humana [4].

Otra variante que aparece es la de los *Self-Organizing Maps* (SOM). Esta estructura puede trabajar con diferentes nociones de distancia entre descriptores, escala linealmente con respecto a la dimensión de los mismos y se caracteriza por realizar una compresión del espacio de entrada, permitiendo trabajar con bases de datos de gran tamaño.

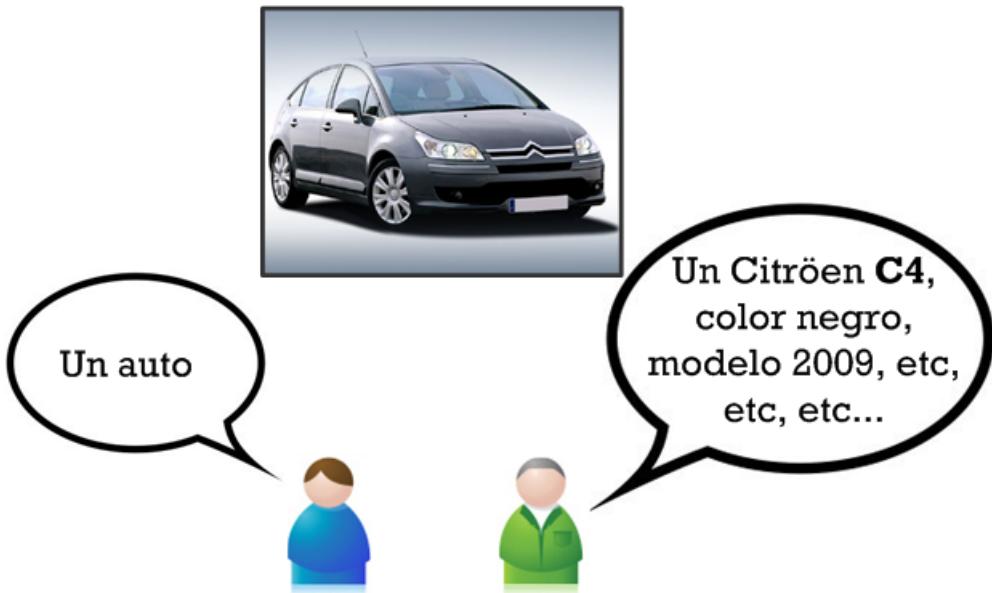


Figura 2.5: Problemas lenguaje natural: Imprecisión

2.4. TBIR

2.4.1. Extracción de características

En los sistemas de TBIR, las imágenes son descriptas a través de un conjunto de palabras. Por cuestiones inherentes al lenguaje natural, la descripción de una imagen puede ser imprecisa y ambigua (Fig. 2.5 y 2.6). Además, surge la necesidad de definir previamente el idioma con el que se va a trabajar, siendo el inglés el más utilizado.

Para extraer información a partir del texto asociado a cada imagen, se realiza un preprocesamiento que puede consistir en la eliminación de palabras muy comunes (conocidas en la literatura como *stopwords*), normalización de caracteres (por ejemplo, convertir todo a minúscula), entre otros. Este preprocesamiento dará lugar a *términos* (*terms*) que se emplearán finalmente para describir a la imagen.

El descriptor que se usará para representar a las imágenes típicamente consiste en una lista de términos y la correspondiente frecuencia de cada uno. Es importante destacar que aquí se ignora el orden exacto de los términos, viendo al texto como una bolsa de palabras (*bag of words*).

A continuación, presentaremos algunas de las tareas más comunes relacionadas al preprocesamiento de texto [25].

Eliminación de *stopwords*

Se conoce como *stopwords* a palabras que son muy comunes dentro del lenguaje o que semánticamente no sirven para discriminar entre descripciones textuales. De esta forma, solamente se almacenan términos relevantes para las búsquedas y, al mismo tiempo, se ahorra

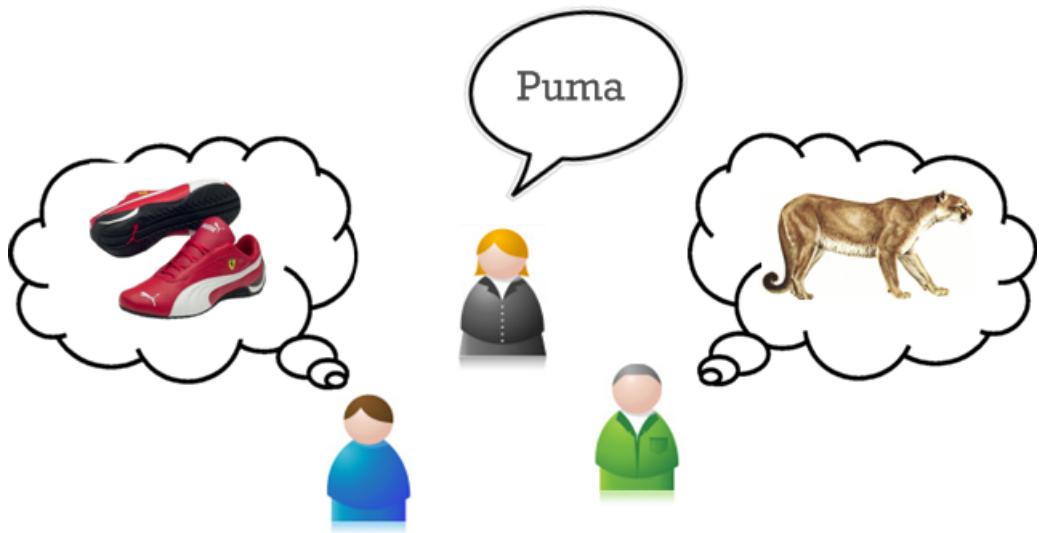


Figura 2.6: Problemas lenguaje natural: Ambigüedad

espacio de almacenamiento en los descriptores. Suele implementarse utilizando una lista de *stopwords* fija.

Normalización de caracteres

La normalización de caracteres consiste en armar clases de equivalencia de términos, haciendo que términos similares vayan a la misma clase de equivalencia. El objetivo es que términos con diferencias superficiales en la secuencia de caracteres sean considerados el mismo. Normalizaciones clásicas son la eliminación de acentos, conversión de mayúsculas a minúsculas, entre otras.

Stemming

La técnica de *stemming* consiste en extraer la raíz de las palabras. Por ejemplo, palabras como “reader” y “reading” serán preprocesadas convirtiéndose ambas en “read”. Las técnicas de *stemming* suelen usar heurísticas para remover parte del final de las palabras con la esperanza de obtener la raíz de la palabra en la mayoría de los casos. El algoritmo más popular de *stemming* es el de *Porter*, definido para el idioma inglés.

2.4.2. Indexación

Sin lugar a dudas, la recuperación basada en texto se encuentra en una etapa mucho más madura y avanzada en comparación con la de recuperación por contenido. En TBIR, la estructura más popular en la actualidad que se utiliza para indexar texto es la de *índices invertidos* [25].

Básicamente, un índice invertido consiste en un diccionario de términos (conocido como *lexicon*), donde cada uno de los términos tiene asociado una lista de los documentos (en este caso, imágenes) en las que aparece el término. Además, junto a cada documento se almacena

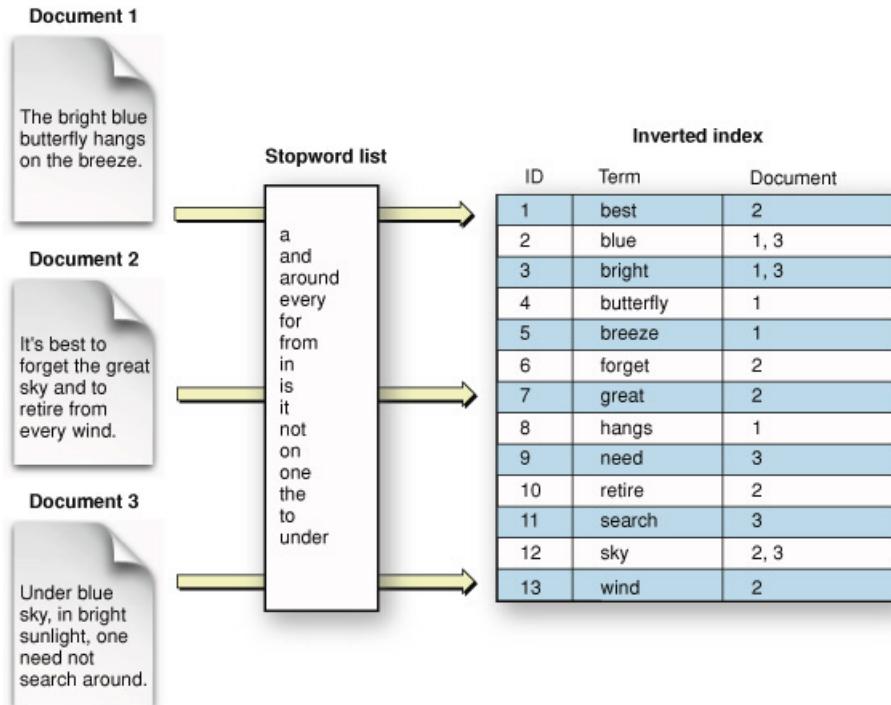


Figura 2.7: Índice invertido (<http://developer.apple.com>)

la frecuencia de ese término en el documento (Fig. 2.7).

Junto a la estructura de índices invertidos, suele utilizarse el modelo de asignación de pesos conocido como *tf-idf* (*term frequency - inverse document frequency*) [25]. *Term frequency* se nota como $tf_{t,d}$ y representa la cantidad de veces que el término t aparece en el documento d . Por otro lado, considerando a N como la cantidad total de documentos de la colección y a df_t como la cantidad de documentos donde aparece el término t , se define a *inverse document frequency* de la siguiente forma:

$$idf_t = \log \frac{N}{df_t}$$

La idea de este modelo de asignación de pesos es darle mayor importancia a los términos que aparecen mucho dentro de un documento, pero atenuando su injerencia cuando son términos muy frecuentes dentro de la colección.

Al igual que sucede con la generación de índices en CBIR, los índices invertidos se construyen en forma *offline*. El algoritmo que se encarga de construir esta estructura permite dividir el trabajo entre varios nodos de procesamiento. De esta forma, se pueden paralelizar las tareas, dándole una gran escalabilidad a esta variante.

2.5. Recuperación

Para la recuperación de imágenes, es posible formular consultas utilizando tanto texto como imágenes. Si la consulta consiste exclusivamente en texto, se aplican las técnicas de TBIR;

mientras que si la consulta está formada únicamente por una imagen, se utiliza CBIR. Ahora, si el usuario provee texto e imagen, el sistema debe encargarse de combinar los resultados de TBIR y CBIR, existiendo diferentes variantes para llevar esto a cabo. Un enfoque consiste en ordenar los resultados de TBIR usando los resultados de CBIR, mientras otras variantes se centran en mezclar ambas listas de resultados (permitiendo asignarle diferente peso a los resultados de TBIR y CBIR) [4].

Durante la etapa de recuperación de imágenes, la participación del usuario juega un papel preponderante. En la recuperación *automática*, el usuario se limita a presentar una imagen y/o texto como consulta y es el sistema quien se encarga por completo de buscar las imágenes más relevantes y mostrar los resultados. Por otro lado, en la recuperación *interactiva*, el usuario se involucra en el proceso de búsqueda de imágenes relevantes. El método más popular dentro de los interactivos es conocido como *Relevance Feedback* y consiste en los siguientes pasos:

1. El usuario presenta una imagen y/o texto como consulta.
2. El sistema devuelve un conjunto de imágenes como resultado inicial.
3. El usuario marca como relevante, no relevante o neutra a cada una de las imágenes devueltas.
4. El sistema calcula un nuevo conjunto de imágenes relevantes en base al *feedback* del usuario y devuelve este nuevo conjunto.
5. Este proceso puede realizar varias iteraciones hasta que el usuario quede conforme con lo devuelto.

En la actualidad, existe una tendencia a evitar involucrar en el proceso de recuperación al usuario, haciendo más fácil y cómodo el uso del sistema. En este trabajo nos centraremos en la recuperación automática.

2.6. Evaluación

La gran cantidad de alternativas que existen a la hora de construir un sistema de VIR vuelve necesario establecer criterios para comparar diferentes implementaciones. Medir la calidad de un sistema es fundamental para satisfacer las necesidades del usuario ya que no sólo es importante la eficiente recuperación de los resultados, sino también que las imágenes obtenidas sean de utilidad.

En el área, pocos trabajos se han dedicado a comparar sistemas de VIR. En los últimos años se han organizado conferencias como *Benchathlon* e *ImageCLEF*, dedicadas a comparar diferentes métodos existentes. Sin embargo, el impacto que éstas han tenido en el área todavía no es el esperado. Al mismo tiempo, una gran cantidad de trabajos publicados en el área utilizan únicamente bases de datos y métricas propias. Este enfoque no sólo dificulta la comparación entre sistemas sino que es simple seleccionar datos y medidas para los cuales la implementación propia muestre mejores resultados.

Características	ZuBuD	UCID	UK Bench	ImageCLEF 2007
Tipo	CBIR	CBIR	CBIR	TBIR + CBIR
Cantidad de imágenes	1.005	1.338	10.200	20.000
Cantidad de consultas	105	262	2.550	60
Cantidad de imágenes por consulta	5	2.5 (promedio)	3	57 (promedio)
Formato imágenes	PNG	TIF	JPG	JPG
Tamaño imágenes	640x480 y 320x240	512x384 y 384x512	640x480	360x480 y 480x360

Tabla 2.1: Información detallada de las BDs de imágenes



Figura 2.8: El mismo edificio desde diferentes perspectivas (imágenes de la base ZuBuD)

A continuación, presentaremos una serie de bases de datos de imágenes de gran utilidad para comparar sistemas y luego mencionaremos algunas de las métricas más comunes para medir la calidad de sistemas de VIR.

2.6.1. Bases de datos de imágenes

Actualmente, la cantidad de bases de datos de imágenes disponibles para comparar sistemas de VIR es escasa y muchas de ellas no son de libre acceso. Aquí se realizará una descripción de cada una de las bases de datos que se usarán a lo largo de este trabajo (información detallada en la Tabla 2.1). Cabe destacar que todas estas bases son de acceso libre y gratuito.

ZuBuD

ZuBuD (*Zurich Buildings Database* [11]) es una base de datos creada por el *Swiss Federal Institute of Technology*. Cuenta con imágenes de edificios de la ciudad de Zurich, tomadas desde diferentes ángulos y bajo diferentes condiciones de luminosidad. La base de datos posee un conjunto de juicios de relevancia (cada uno formado por una imagen de consulta más una lista de las imágenes esperadas como resultado), donde las imágenes relevantes son las que se refieren al mismo edificio (Fig. 2.8) y se orienta a evaluar sistemas de CBIR.

UCID

La base de datos UCID (*Uncompressed Colour Image Database* [12]) fue creada por la *Universidad de Nottingham*, Gran Bretaña. Su objetivo fue armar un conjunto de imágenes estándar a la hora de comparar sistemas de CBIR así como también aplicaciones de compresión de imágenes.

UCID está formada por imágenes a color de diferentes temáticas. Además, cuenta con juicios de relevancia que fueron realizados manualmente. Las imágenes que son consideradas relevantes son claramente similares para el ojo humano, sin embargo en la base existen otras

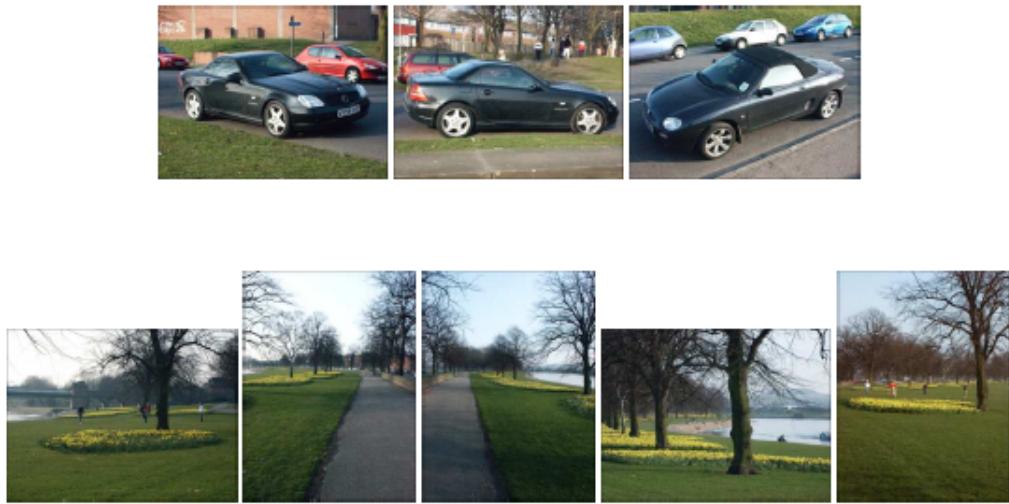


Figura 2.9: Imágenes de la base UCID

imágenes que tal vez son visualmente similares pero no se consideran relevantes. Ejemplos de las imágenes pueden verse en la Fig. 2.9.

UK Bench

UK Bench[13] es una base de datos orientada a reconocimiento de objetos y puede utilizarse para evaluar sistemas de CBIR. Fue desarrollada por la *Universidad de Kentucky* y cuenta con imágenes de objetos de uso cotidiano. Cada objeto cuenta con 4 imágenes tomadas desde diferentes ángulos y bajo distintas condiciones de luminosidad. En la Fig. 2.10 aparecen ejemplos de imágenes presentes en la base de datos.

ImageCLEFphoto 2007

Durante la competencia realizada en el año 2007 y organizada por la ImageCLEF, se utilizó la base de datos *IAPR TC-12 Benchmark*[14]. Cuenta con imágenes y con un texto descriptivo para cada una de ellas, disponible en varios idiomas (en este trabajo sólo usaremos las descripciones en inglés). Las imágenes corresponden a fotografías tomadas en diferentes partes del mundo y abarcan diversas temáticas. Ejemplos de imágenes pueden observarse en la Fig. 2.11.

2.6.2. Métricas

Al momento de comparar sistemas de VIR, surge la necesidad de encontrar métricas que midan la calidad de los sistemas. Dependiendo de la aplicación o contexto donde sea utilizado el sistema, los requerimientos de los usuarios pueden variar y es difícil encontrar una sola métrica que se ajuste a todos los casos. En el área no existe un consenso acerca de cuál es la métrica más adecuada, pero sí conviven una serie de medidas que pueden considerarse estándar para analizar la performance de los sistemas de VIR. A la hora de definir y usar una métrica,

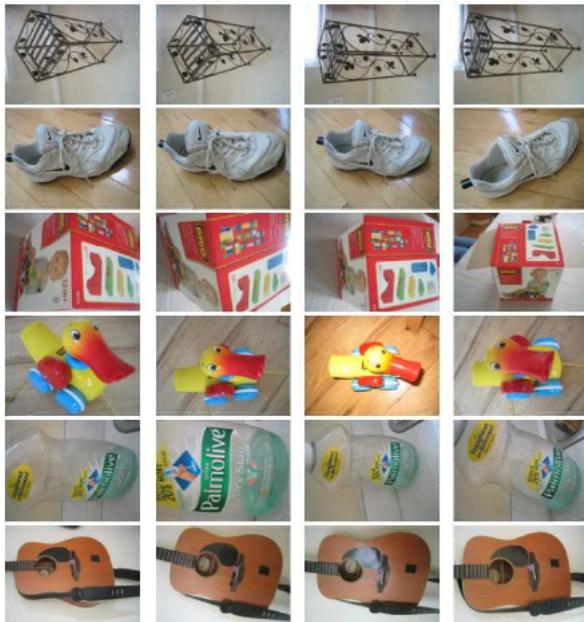


Figura 2.10: Imágenes de la base UK Bench



Figura 2.11: Imagen de la ImageCLEF 2007 junto a su texto asociado

Nombre	Descripción
N_H	Cantidad de resultados devueltos (<i>hitlist</i>)
$H[i]$	La imagen i de la <i>hitlist</i>
$Rel[i]$	Es 1 si $H[i]$ es relevante; si no, 0
N_R	Cantidad de imágenes relevantes
Q	Conjunto de consultas
N_Q	Cantidad de consultas
P	Precisión
R	Recall

Tabla 2.2: Variables utilizadas para definir las métricas

hay dos puntos importantes que deben tenerse en cuenta: por un lado, que sea una medida fácil de interpretar y, por otro lado, que intente capturar la noción de calidad de los usuarios del sistema. En las siguientes secciones presentaremos algunas de las métricas más utilizadas en la actualidad [25] (en la Tabla 2.2 se incluyen variables comunes para las definiciones de las métricas).

Precisión y Recall

Dos de las métricas más conocidas en la literatura son la *precisión* y el *recall*. La *precisión* representa la proporción de imágenes recuperadas que son relevantes. El *recall* representa la proporción de imágenes relevantes que son recuperadas. Formalmente,

$$\text{Precision}(q) = \frac{1}{N_H} \sum_{i=1}^{N_H} Rel[i] = \frac{\#\text{Imágenes relevantes recuperadas}}{\#\text{Imágenes recuperadas}}$$

$$\text{Recall}(q) = \frac{1}{N_R} \sum_{i=1}^{N_R} Rel[i] = \frac{\#\text{Imágenes relevantes recuperadas}}{\#\text{Imágenes relevantes}}$$

Una ventaja de contar con dos métricas que evalúen diferentes cuestiones es que, en ciertas aplicaciones, a una de ellas se le puede asignar más importancia que a la otra.

Por último, estas métricas están definidas para una sola consulta, pero cuando se habla de la *precisión* o *recall* del sistema, se hace referencia al promedio de esos valores entre todas las consultas.

F-Measure

La precisión y el recall pueden ser combinadas en una sola medida: *F-Measure*. La idea de esta medida es unir la precisión y el recall en un solo valor, penalizando a sistemas en los que alguna de estas dos medidas sea baja. Es una opción interesante para contextos donde tanto la precisión como el recall son importantes.

$$F = \frac{2PR}{P + R}$$

Precision at k

En muchas aplicaciones como búsquedas en la *Web*, lo más importante es saber qué tan buenos son los primeros resultados. Con esta idea en mente, se define la métrica *Precision at k*, donde se calcula la precisión considerando únicamente los primeros k resultados. Valores típicos para k son 10 o 20.

$$\text{Precision-at-}k(q) = \frac{1}{k} \sum_{i=1}^k \text{Rel}[i]$$

Mean Average Precision (MAP)

MAP es una medida de una sola dimensión que mide la performance con respecto a la recuperación de las imágenes relevantes. Premia a aquellos sistemas que devuelven las imágenes relevantes en las primeras posiciones del resultado (con rankings altos). Es el promedio de la precisión cada vez que una imagen relevante es encontrada (si una relevante no es recuperada por el sistema, su precisión se considera 0).

$$\text{MAP} = \frac{1}{N_Q} \sum_{q \in Q} \text{AP}(q)$$

$$\text{AP}(q) = \frac{1}{N_R} \sum_{i=1}^{N_H} \text{Precision-at-}i(q) \times \text{Rel}[i]$$

Por ejemplo, si una consulta tiene 4 documentos relevantes que son recuperados en las posiciones 1, 2, 4 y 7. La precisión es 1, 1, 0.75 y 0.57 respectivamente, siendo AP igual a 0.83.

A diferencia de las métricas definidas anteriormente, el MAP es poco intuitivo lo que dificulta en gran parte el análisis de los resultados.

2.7. Aplicaciones

Los sistemas de VIR tienen una gran cantidad de aplicaciones y los campos donde se usan son diversos. A continuación presentaremos algunas de las aplicaciones más importantes.

Búsquedas en la Web

El uso de Internet ha crecido exponencialmente en los últimos años y la cantidad de imágenes disponibles allí es cada vez mayor. Tanto a través de texto como de una imagen o usando ambas, es posible recuperar imágenes que se ajusten a lo ingresado en la consulta.

Investigación criminal

Cuando una persona es detenida, la Policía suele encargarse de tomarle fotografías así como también sus huellas digitales. Estas imágenes pueden ser archivadas junto con los datos del acusado. Entonces, a partir de imágenes tomadas por cámaras de seguridad o identikit más alguna descripción del acusado, se puede realizar una búsqueda y así lograr reconocer sospechosos con antecedentes.

Medicina

Radiografías, ecografías, tomografías y otros estudios suelen practicarse sobre los pacientes con el fin de realizar diagnóstico y monitoreo. Estas imágenes en general se asocian a la historia clínica de cada paciente. Al momento de realizar un nuevo estudio, éste puede utilizarse para compararlo con estudios anteriores del mismo paciente o de otros pacientes para comparar su evolución y realizar un diagnóstico más preciso.

Detección de robo de imágenes

En muchas bases de datos de imágenes comerciales, los clientes están habilitados para ver las imágenes pero si quieren hacer uso de ellas, deben pagar. Las compañías se encuentran con el problema de detectar a usuarios que tomaron una foto de la base de datos, la modificaron y comenzaron a utilizarla sin previo pago. Aquí es posible utilizar CBIR de la siguiente forma: las imágenes de la base de datos original son sometidas a modificaciones como reducción de la calidad, recortado (*cropping*), distorsión del color o combinaciones de ellas; luego, se usa esa base de datos con las imágenes modificadas para realizar búsquedas con el fin de detectar posibles plagios.

Capítulo 3

Histograma de Color

El color es una de las características más intuitivas a la hora de describir una imagen. Uno de los descriptores más populares dentro del área de CBIR se basa en esta propiedad y es conocido como *histograma de color*. A lo largo de este capítulo detallaremos sus principales características y su aplicación a sistemas de CBIR.

3.1. Descripción general

Uno de los descriptores más simples e intuitivos es el histograma de color. Este descriptor fue introducido por Swain y Ballard [24] y ha sido utilizado intensamente en trabajos del área [2, 27, 31]. La idea consiste en contar la cantidad de píxeles de cada color presentes en una imagen, armando un histograma que representa las frecuencias con que cada color aparece en la imagen.

Por lo general, las imágenes digitales son almacenadas en formato RGB (Red-Green-Blue) y se utilizan 8 bits para cada componente, generando una paleta de $(2^8)^3 = 256^3 = 16.7$ millones de colores. Sin lugar a dudas, este alto número de colores hace necesario trabajar con paletas más pequeñas que den lugar a histogramas más compactos. La idea será reducir la paleta de colores, agrupando colores perceptualmente similares. Este proceso se conoce como *cuantización de los colores*. La cantidad de colores considerados determinará la cantidad de *bins* que tendrá el histograma. Además, deberá definirse una medida que sirva para decidir cuándo dos histogramas son similares. Por último, se buscará definir una función de *scoring* que permita ordenar los resultados en base a su puntaje y, eventualmente, eliminar aquellos resultados con puntajes bajos.

Concretamente, las siguientes cuestiones deberán definirse para trabajar con histogramas de color:

- Espacio de color
- Cuantización de los colores
- Cantidad de *bins* del histograma
- Función de distancia entre histogramas

- Función de *scoring* para histogramas

En las secciones subsiguientes nos dedicaremos a estudiar en profundidad el uso de histogramas de color para el área de CBIR.

3.2. ¿Por qué utilizar Histogramas de Color en CBIR?

En la actualidad, diversos tipos de descriptores aparecen como candidatos para describir visualmente a las imágenes. Muchos sistemas de CBIR utilizan descriptores complejos o una combinación de varios descriptores, convirtiendo al sistema en una *caja negra* donde se vuelve difícil determinar la vinculación entre la consulta y lo que el sistema devuelve. Como mencionamos previamente, una de las características más intuitivas que tienen las imágenes es el color. Por este motivo, usar una característica intuitiva y fácil de comprender por el usuario común transforma al histograma de color en un candidato interesante.

En términos de complejidad temporal y espacial, los histogramas de color tienen un buen desempeño. El algoritmo para calcularlos es simple, por lo que no necesitan una gran capacidad de cómputo. Además, la cantidad de *bins* no suele ser muy alta, haciendo que el espacio requerido sea bajo.

Debido a que los histogramas representan la distribución del color, es importante destacar que son invariantes a cambios de escala de imágenes, a rotaciones y a efectos de espejado (*mirroring*).

3.3. Espacios de Color

Un espacio de color determina la forma en la cual se representarán los diferentes colores. En la actualidad, existe una gran cantidad de espacios de color pero, sin lugar a dudas, el RGB es el más popular. En gran parte, esto se debe a que la mayoría de los dispositivos electrónicos como televisores o monitores se basan en la adición de rojo, verde y azul para representar a los colores.

Si bien el RGB es el espacio más utilizado, se han desarrollado variantes que describen a los colores en una forma diferente. Surgen a partir de la observación de que los usuarios típicos no suelen pensar al color en términos de rojo, verde y azul. Uno de los ejemplos de esta tendencia es el espacio *HSV* (Hue-Saturation-Value), ya que define a los colores en base al tono o tinte (*hue*), cantidad de color presente (*saturation*) y brillo (*value*).

3.3.1. RGB

El espacio de color RGB utiliza a los colores primarios rojo, verde y azul en forma aditiva para representar a los colores. Típicamente, se utilizan 8 bits para cada componente, permitiendo valores entre 0 y 255. En su forma más general, puede asumirse que las componentes R, G y B toman valores entre 0 y 1. De esta forma, el espacio de color RGB puede verse como un cubo (Fig. 3.1).

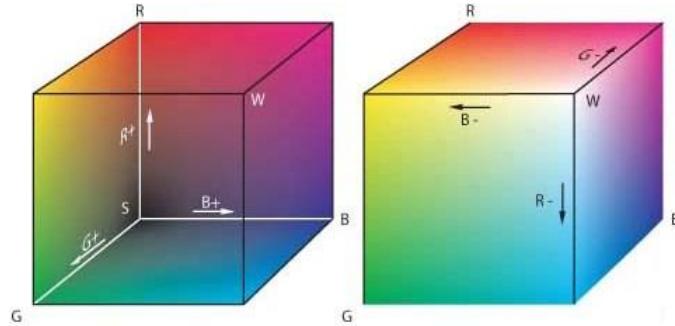


Figura 3.1: Espacio de color RGB

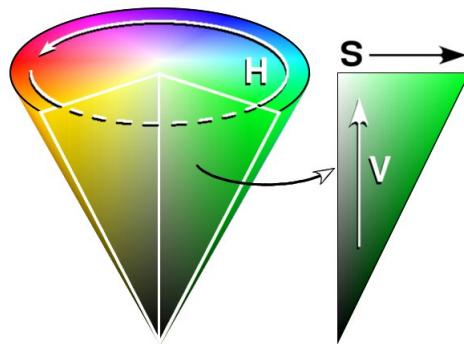


Figura 3.2: Espacio de color HSV

A pesar de su extenso uso, en ciertas aplicaciones el espacio RGB no es el más adecuado ya que no es perceptualmente uniforme. Decimos que un espacio de color es *perceptualmente uniforme* cuando una pequeña perturbación en las componentes de un color se traduce en una modificación aproximadamente igual en cómo uno percibe el color. Por ejemplo, el concepto anterior es utilizado en las radios donde un giro leve de la rueda del volumen se traduce en una leve modificación del nivel de sonido percibido.

3.3.2. HSV

El espacio de color HSV intenta representar a los colores de forma intuitiva, utilizando para ello el tono (*hue*), la cantidad de color presente (*saturation*) y el brillo (*value*). La componente *hue* suele medirse en grados (de 0° a 360°) y las de *saturation* y *value* en valores entre 0 y 1. El espacio generado por HSV puede verse como un cono (Fig. 3.2): el eje vertical representa el brillo (de oscuro a claro), el eje horizontal es la saturación y el ángulo alrededor del eje horizontal, el tono.

Una de sus principales características es que es perceptualmente más uniforme que el espacio RGB [32, 33]. Además, es sencilla la conversión desde el espacio RGB hacia el HSV, lo que lo vuelve interesante ya que la mayoría de las imágenes están en formato RGB. Además, si se desea utilizar para histogramas de color, el tener componentes intuitivas permite, por ejemplo, darle mayor importancia a la componente del color que a las de saturación o brillo.

3.4. Cuantización de los colores

La cuantización de los colores consiste en, a partir de una paleta original de N colores, agrupar colores perceptualmente similares formando una paleta reducida de n colores donde $N >> n$. La cantidad de colores de la paleta reducida determinará la cantidad de *bins* que tendrá el histograma.

Típicamente, se distingue entre cuantizaciones *uniformes* y *no-uniformes*. En la uniforme, cada grupo de colores tiene el mismo tamaño [32]; mientras que en la no-uniforme, puede haber grupos de diferente tamaño [33].

3.5. Cantidad de *bins* del histograma

Determinar el número de *bins* para un histograma es una decisión crucial. Por un lado, se intentará contar con los *bins* suficientes como para que los resultados obtenidos sean de calidad y, por otro lado, se buscará utilizar la menor cantidad posible de *bins* para que el espacio ocupado sea menor y se facilite la etapa de indexación de los histogramas.

Como mencionamos anteriormente, la cuantización elegida determinará la cantidad de *bins* del histograma. Es claro que contar con muchos *bins* hará que imágenes similares sean clasificadas como diferentes (colores parecidos irán a *bins* distintos), mientras que usar pocos *bins* hará lo contrario: considerará a imágenes diferentes como parecidas (colores muy distintos irán al mismo *bin*). Por este motivo, la cantidad de *bins* a utilizar juega un papel importante en cuanto a la calidad de los resultados.

En lo posible, el número de *bins* empleados tratará de mantenerse bajo. De esta forma, los histogramas ocuparán menos espacio y además será más fácil indexarlos.

Para histogramas en RGB, es común utilizar 512 *bins* (8x8x8, cuantización uniforme de 8 niveles por componente) [31] y, para histogramas en HSV, no hay un consenso acerca de la cantidad de *bins* pero se suele asignar más niveles a la componente *hue*.

3.6. Función de distancia entre histogramas

Una vez construidos los histogramas, es necesario utilizar una función que determine cuándo dos histogramas deben considerarse similares. Una de las métricas más utilizadas y una de las que mejores resultados ha dado en la literatura [31] es la norma *L1*.

$$\mathbf{d}_{\mathbf{L1}}(H, H') = \sum_{i=1}^n |H[i] - H'[i]|$$

3.7. Función de *scoring* para histogramas

A diferencia de lo que sucede en TBIR, la definición de una función de scoring no ha recibido gran atención en el área de CBIR. Contar con una función de scoring adecuada no sólo permite ordenar los resultados en base a su puntaje sino también abre la posibilidad de

descartar a aquellos resultados con puntajes bajos, probablemente poco interesantes para la búsqueda.

Antes de entrar en mayores detalles, vale la pena comparar la recuperación en TBIR y en CBIR para comprender las dificultades adicionales que se presentan en búsquedas por contenido visual. Por ejemplo, si en un sistema de TBIR la consulta es la palabra “casa”, se recuperarán exclusivamente imágenes en cuya descripción aparezca esa palabra. Es decir, imágenes que no contengan en su descripción a “casa” directamente no aparecerán en los resultados. Ahora, si en CBIR se utiliza como consulta a una imagen de una casa, sin dudas poder determinar si una imagen contiene o no a una casa será una tarea mucho más compleja que en TBIR.

Típicamente, cuando un sistema de CBIR que utiliza histograma de color recibe una consulta, se computa su histograma y se lo compara contra los histogramas precomputados de las imágenes de la colección. A la hora de comparar los histogramas, se utiliza una función de distancia que le indica qué tan cercanas se encuentran ambas imágenes. Luego, para presentar los resultados, se puede ordenarlos de menor a mayor distancia, mostrando primero los más cercanos a la imagen de consulta. Sin embargo, ¿hasta qué distancia tiene sentido presentar los resultados? ¿A partir de qué valor de distancia los resultados corresponden a imágenes muy diferentes?

Frente a este escenario, una alternativa podría ser fijar un valor k arbitrario y devolver las k primeras imágenes. No obstante, puede ser complicada la elección del k y se otorga poca flexibilidad para consultas que tienen menos o más de k imágenes relevantes. Por lo tanto, proponemos la siguiente idea: permitir fijar un *umbral* y descartar aquellas imágenes que se encuentren por debajo de ese valor (ver Figura 3.3). Para poder aplicar este concepto, una condición necesaria es que las distancias entre imágenes se encuentren normalizadas y esto depende exclusivamente de la función de distancia utilizada. Para el caso de la norma L1, las distancias no están normalizadas pero es sencillo llevarlas al intervalo $[0, 1]$ (demonstración en el Anexo A). Además, para utilizar el mismo razonamiento que en sistemas de TBIR donde los puntajes de los resultados se ordenan de mayor a menor, se puede restar esta distancia normalizada a 1, dando lugar a la siguiente función de scoring:

$$\text{score}(H_{\text{query}}, H) = 1 - \text{normalizar}(\mathbf{d}_{\mathbf{L1}}(H_{\text{query}}, H))$$

3.8. Construcción del histograma

A lo largo de las secciones anteriores, tratamos diferentes cuestiones que deben ser definidas antes de empezar a trabajar con los histogramas de color. Ahora, vamos a dedicarnos a describir *detalladamente* cómo se construye un histograma.

Lo primero que debe considerarse es el espacio de color utilizado. Si queremos trabajar con RGB, como es el formato estándar de las imágenes, no deberá realizarse ninguna conversión de los píxeles de la imagen. En cambio, si queremos trabajar con HSV, cada píxel deberá ser convertido, utilizando el Algoritmo 1 [34].

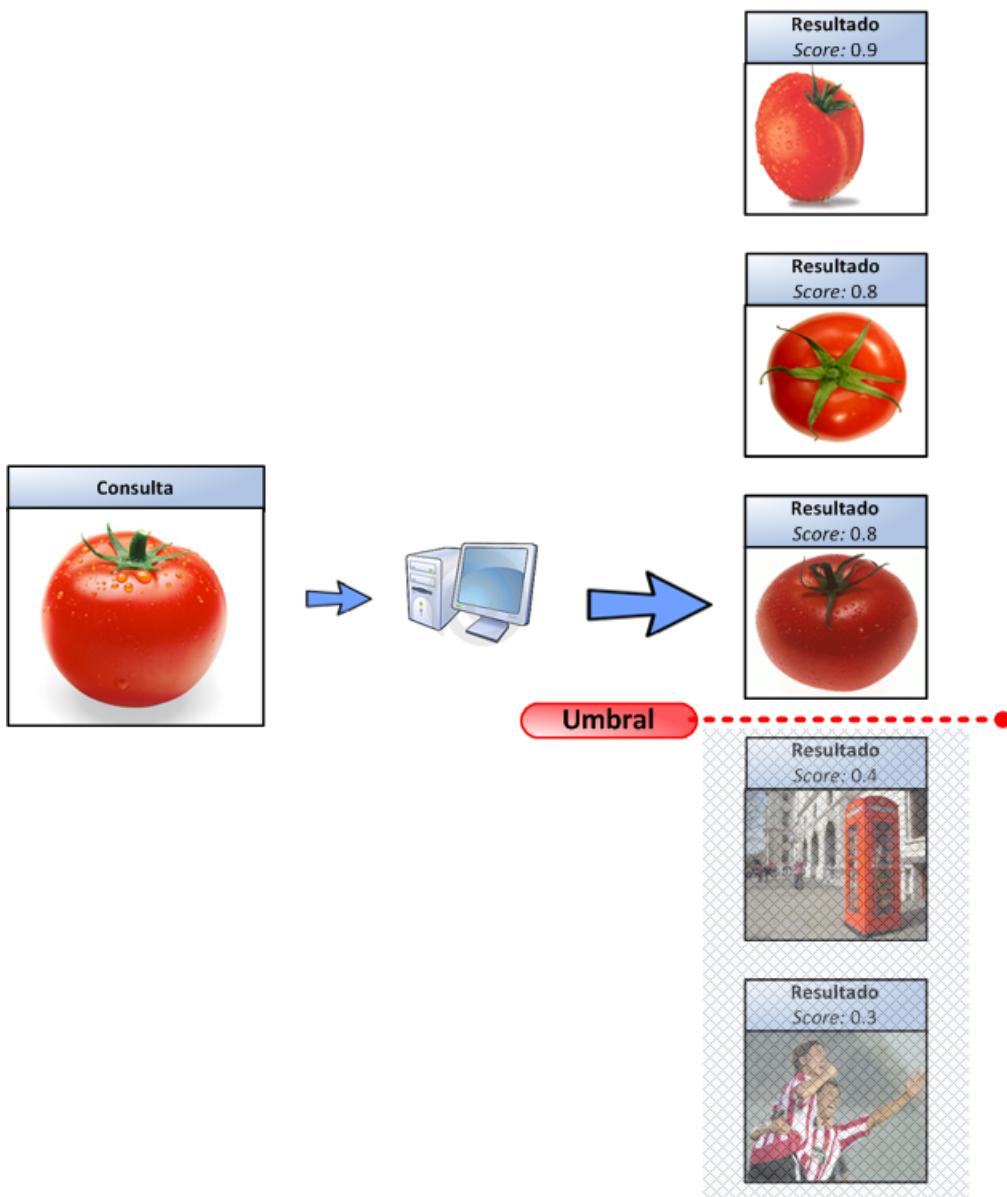


Figura 3.3: Ejemplo de uso de umbral para limitar la cantidad de resultados

Algoritmo 1 Conversión de RGB a HSV

Obs1: Asume que R, G y B son valores normalizados, en el intervalo [0,1]

Obs2: Devuelve H en el intervalo [0,360] y S, V en [0,1]

convertirRGBaHSV(*Double R, Double G, Double B*)

Inicio

 max $\leftarrow \max(R, G, B)$

 min $\leftarrow \min(R, G, B)$

 V $\leftarrow \max$

Si max = 0

 S $\leftarrow 0$

Si no

 S $\leftarrow (\max - \min) / \max$

Si max = min

 H $\leftarrow 0$ //Acromático

Si max = R y G $\geq B$

 H $\leftarrow (60 * (G - B)) / (\max - \min)$

Si max = R y G $< B$

 H $\leftarrow 360 + (60 * (G - B)) / (\max - \min)$

Si G = max

 H $\leftarrow 60 * (2 + (B - R)) / (\max - \min)$

Si no

 H $\leftarrow 60 * (4 + (R - G)) / (\max - \min)$

Fin

Una vez realizadas las conversiones correspondientes, se procede a construir el histograma según el Algoritmo 2. La idea es iterar por los píxeles, determinando a qué *bin* deben ir (dependiendo de la cuantización usada) e incrementando la cantidad de píxeles en esa posición. En el caso de cuantización uniforme, se decide el *bin* correspondiente usando el Algoritmo 3; mientras que en el caso de la no-uniforme, existen muchas variantes y depende de la función de cuantización que se elija. Al terminar de recorrer la imagen, se normaliza el histograma para que sea invariante a cambios de escala, utilizando el Algoritmo 4.

Algoritmo 2 Construcción del histograma

construirHistograma(*Imagen* imagen)

Inicio

Para cada pixel de imagen

 bin \leftarrow **cuantizar**(pixel)

 histograma[bin]++

Fin para

normalizar(histograma, imagen.cantidadDePixel)

Fin

Algoritmo 3 Cuantización uniforme de píxeles

Obs: Asume que las componentes de cada píxel están normalizadas, en el intervalo [0,1)

cuantizar(*Pixel* pixel)

Inicio

 X \leftarrow pixel[0]

 Y \leftarrow pixel[1]

 Z \leftarrow pixel[2] //X,Y,Z dependerán del espacio de color usado

 //Considerando a n_X, n_Y, n_Z como la cantidad de niveles de cuantización de cada canal
 bin $\leftarrow \lfloor n_X \times X \rfloor \times n_Y \times n_Z + \lfloor n_Y \times Y \rfloor \times n_Z + \lfloor n_Z \times Z \rfloor$

Fin

Algoritmo 4 Normalización de histograma

normalizar(*Histograma* histograma, *Int* cantidadTotalDePixel)

Inicio

Para cada bin $\leftarrow 1 \dots$ histograma.cantidadDeBins

 histograma[bin] \leftarrow histograma[bin] / cantidadTotalDePixel

Fin para

Fin

3.9. Experimentación

3.9.1. Descripción

En esta sección analizaremos diferentes variantes asociadas a los histogramas de color. Se realizarán experimentos con los espacios de color RGB y HSV y se los evaluará mediante la norma L1.

Para el espacio RGB se utilizará cuantización uniforme dividiendo cada canal en 8 niveles ($8 \times 8 \times 8$), obteniendo un histograma de 512 *bins*. Para el espacio HSV, se usará también cuantización uniforme, pero dándole mayor importancia a la componente *Hue* ($32 \times 4 \times 4 = 512$ *bins*).

Los experimentos se realizarán sobre las bases de imágenes presentadas en la sección 2.6.1: *ZuBuD*, *UCID* y *UK Bench*. Consistirán en analizar el comportamiento de los histogramas para diferentes umbrales y se usarán las métricas *MAP*, *Precision*, *Recall* y *F-Measure* (descriptas en la sección 2.6.2). Para cada consulta, se considerarán como máximo 1.000 resultados (para evitar devolver la base de imágenes entera, en casos de umbrales bajos).

3.9.2. Resultados

Umbral	ZuBuD				UCID				UK Bench			
	MAP	Prec	Recall	F	MAP	Prec	Recall	F	MAP	Prec	Recall	F
0.0	75,95	0,50	100,00	0,99	39,77	0,24	98,42	0,48	61,34	0,28	93,03	0,56
0.1	75,95	0,50	100,00	0,99	39,77	0,24	98,04	0,48	61,34	0,28	93,03	0,56
0.2	75,95	0,51	100,00	1,01	39,77	0,28	98,04	0,56	61,34	0,30	92,99	0,60
0.3	75,95	0,55	100,00	1,10	39,76	0,47	95,68	0,93	61,34	0,46	92,70	0,88
0.4	75,95	0,73	99,83	1,46	39,69	1,79	88,73	2,96	61,32	1,32	91,61	2,09
0.5	75,94	1,44	97,91	2,82	38,49	6,34	72,85	7,76	61,13	4,96	87,88	6,80
0.6	75,76	5,95	95,30	10,15	35,60	13,07	54,67	15,31	59,66	16,92	79,11	19,76
0.7	73,32	31,82	85,22	36,20	26,37	22,53	29,95	21,07	52,01	40,07	61,66	38,09
0.8	55,41	70,29	57,39	57,20	10,20	16,89	10,52	11,48	31,08	44,04	33,12	33,55
0.9	14,26	40,87	14,26	20,17	1,08	1,34	1,08	1,08	5,66	12,99	5,70	7,62

Tabla 3.1: RGB 8x8x8

Umbral	ZuBuD				UCID				UK Bench			
	MAP	Prec	Recall	F	MAP	Prec	Recall	F	MAP	Prec	Recall	F
0.0	84,15	0,50	100,00	0,99	52,33	0,24	97,94	0,48	69,36	0,28	94,84	0,57
0.1	84,15	0,51	100,00	1,02	52,33	0,26	97,94	0,53	69,36	0,29	94,82	0,58
0.2	84,15	0,61	100,00	1,21	52,32	0,49	95,86	0,96	69,36	0,37	94,63	0,74
0.3	84,14	1,05	99,13	2,05	52,24	1,61	87,77	2,90	69,35	0,93	93,58	1,72
0.4	84,12	4,11	97,74	6,25	51,16	6,06	78,54	9,23	69,22	4,28	90,26	6,69
0.5	83,10	11,03	93,39	15,33	48,12	18,92	61,03	22,82	68,13	17,17	84,31	21,24
0.6	81,60	35,82	88,52	42,27	37,53	37,95	41,04	34,04	63,37	40,37	73,42	41,04
0.7	68,13	75,49	69,22	66,98	18,50	26,84	19,22	20,38	49,06	55,58	53,63	46,66
0.8	33,04	69,17	33,04	42,26	3,65	7,63	3,65	4,57	24,35	40,07	25,31	28,00
0.9	1,91	9,56	1,91	3,19	0,16	0,76	0,16	0,26	3,78	8,38	3,80	5,02

Tabla 3.2: HSV 32x4x4

Como vemos en las Tablas 3.1 y 3.2 y en los Gráficos 3.4, 3.5 y 3.6, puede observarse que para umbrales bajos el recall es alto y la precisión es baja (se recuperan muchas imágenes relevantes, pero con una gran cantidad de imágenes no relevantes adicionales, reduciendo la

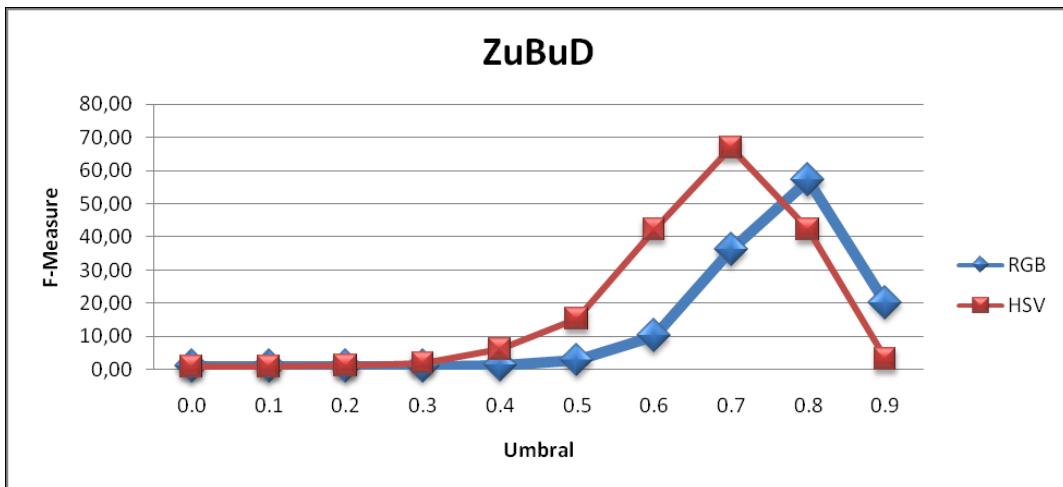


Figura 3.4: Gráfico de F-Measure vs. Umbral para base de datos ZuBuD

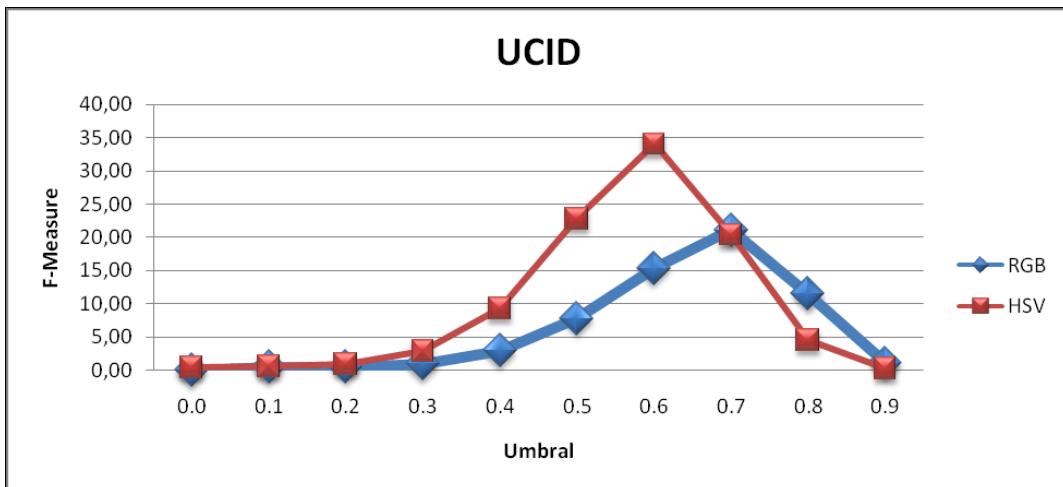


Figura 3.5: Gráfico de F-Measure vs. Umbral para base de datos UCID

precisión). Como es de esperarse, el F-Measure es bajo (penaliza cuando el recall o la precisión son bajos). La métrica MAP es alta a pesar de que la precisión es considerablemente baja. A medida que los umbrales van aumentando (entre 0,6 y 0,8), la precisión va aumentando y el recall disminuyendo pero puede observarse que en las diferentes bases de datos se consigue el mayor equilibrio entre precisión y recall, obteniendo los valores más altos de F-Measure. A pesar del aumento de la precisión, el MAP se reduce mostrando un comportamiento parecido al recall. Para umbrales muy altos, la precisión y el recall bajan (la cantidad de imágenes recuperadas son pocas debido a que tienen que tener un score muy alto para ser devueltas) y el F-Measure vuelve a bajar.

Comparando los espacios de color, se puede observar que HSV mejora considerablemente los resultados de RGB en las 3 bases de datos. Si tomamos los mejores resultados de RGB y HSV en cada una de las bases con respecto a F-Measure, en la ZuBuD se consigue una

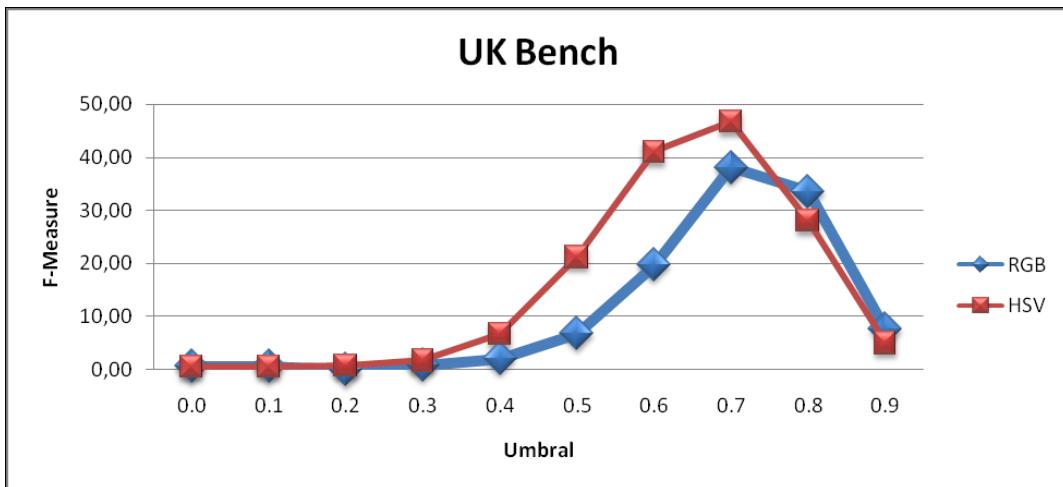


Figura 3.6: Gráfico de F-Measure vs. Umbral para base de datos UK Bench

mejora del 15 %, en la UCID del 38 % y en la UK Bench del 18 %. El hecho de que el espacio HSV sea perceptualmente más uniforme que el RGB hace que la recuperación sea más efectiva.

Capítulo 4

Self-Organizing Maps (SOM)

Las redes neuronales artificiales han sido objeto de estudio desde principios de los años '40. Inspiradas en cómo funcionan las neuronas en el cerebro humano, las redes neuronales son modelos matemáticos que representan una nueva forma de resolver problemas computacionales y se encuadran dentro del paradigma conexiónista. Se destacan principalmente por su gran capacidad de adaptarse a los datos de entrada sin requerir de una programación previa, diferenciándose del clásico paradigma procedural.

La forma en que las redes neuronales son entrenadas no es única y se han desarrollado numerosas variantes como los *Perceptrones*, *Redes ART*, *Redes de Hopfield*, entre otras [5]. Dentro del paradigma del aprendizaje no supervisado, se destacan los *mapas auto-organizados de Kohonen* (Self-Organizing Maps, SOM), que han acaparado gran atención desde su surgimiento en el año 1982 [7]. En este capítulo nos centraremos en describir cómo funciona este modelo y variantes asociadas al mismo.

4.1. Introducción

Los SOM fueron desarrollados por Teuvo Kohonen, profesor de la Universidad de Helsinki, Finlandia. Desde su publicación en el año 1982 [7] hasta el día de la fecha, han alcanzado una gran popularidad y han sido utilizados por diversas disciplinas, aplicándose a tareas que van desde reconocimiento de caracteres hasta el control de un robot.

Este modelo se inspiró en el funcionamiento de la corteza cerebral, donde existen diferentes áreas (visual, táctil, auditiva, entre otras) y los estímulos que se reciben desde el exterior son mapeados al área correspondiente. Además, dentro de cada área, neuronas cercanas responden a estímulos similares. Por ejemplo, dentro de la zona auditiva, neuronas cercanas responderán ante sonidos de similar frecuencia (área conocida como *tonotopic map*).

El algoritmo del SOM está basado en un aprendizaje no supervisado y competitivo. Se denomina *no supervisado* porque el entrenamiento consiste sólo en datos de entrada (algunos modelos como los Perceptrones poseen además la salida esperada de la red) y la red se adapta en base a la experiencia adquirida a través de los patrones presentados previamente. Además, se lo llama *competitivo* debido a que las neuronas de la red compiten entre sí para ver cuál es

activada. A diferencia de otros modelos, una sola neurona es activada al mismo tiempo.

El principal objetivo de los SOM es agrupar patrones de entrada similares. A cada patrón de entrada (en general de alta dimensionalidad), se le asigna una neurona (ubicada en una grilla, comúnmente de dos dimensiones). Esta grilla de neuronas formará un mapa que preserva la topología del espacio de entrada original. De esta forma, elementos cercanos en el espacio de entrada serán mapeados a elementos cercanos en el espacio de salida y, al mismo tiempo, elementos alejados en la entrada serán asociados a elementos alejados en la salida. Además, el mapa generado respeta la densidad de los datos, haciendo que regiones con mayor densidad de elementos en el espacio de entrada reciban mayor cantidad de neuronas que regiones de menor densidad. El hecho de que el mapa sea de baja dimensión lo vuelve una poderosa herramienta para visualizar la distribución original de los datos de entrada.

4.2. Arquitectura

En un SOM, el espacio de entrada está definido por un conjunto de N patrones, cada uno de dimensión m .

$$P = \{x_1, \dots, x_N\}$$

$$x_i = [x_{i_1}, \dots, x_{i_m}]$$

Típicamente, las neuronas se ubican en una grilla rectangular de dos dimensiones ($n_1 \times n_2 = n$). Además, cada neurona tiene asociado un vector de pesos w de dimensión m , igual dimensión que los patrones de entrada (Ver Fig. 4.1).

$$\begin{bmatrix} w_1 \\ \ddots \\ w_n \end{bmatrix}$$

$$w_i = [v_{i_1}, \dots, v_{i_m}]$$

Es importante recordar que la red se entrenará utilizando únicamente los patrones de entrada (aprendizaje no supervisado).

4.2.1. Construcción de un SOM

Para construir un SOM se pueden distinguir básicamente tres etapas:

- Inicialización
- Entrenamiento
- Rotulado

A continuación describiremos en qué consiste cada una de ellas.

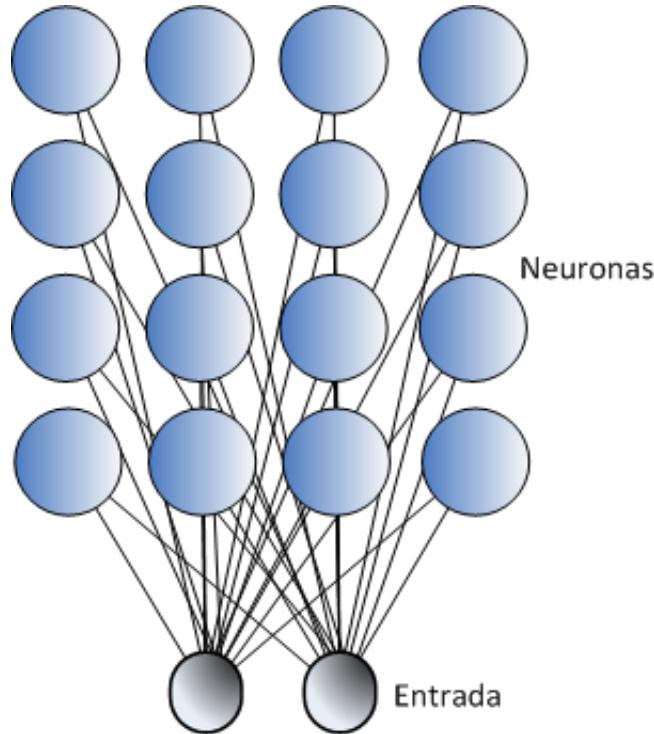


Figura 4.1: Estructura típica de un SOM

Inicialización

El primer paso consiste en asignarle valores iniciales a los vectores de pesos de cada neurona. Una de las opciones más utilizadas es inicializar los pesos con valores aleatorios. Una característica importante del SOM es que justamente puede conseguir un ordenamiento de los datos más allá de que los valores iniciales estén completamente desordenados [8].

Entrenamiento

El entrenamiento de un SOM se realiza por épocas, en forma iterativa (se lo conoce como algoritmo *online*). Esta es la etapa que más costo computacional tiene y durante la misma se van presentando los patrones de entrada al SOM para que éste “aprenda” su distribución. A lo largo de cada época, todos los patrones de entrada son presentados a la red en forma aleatoria, haciendo que el algoritmo sea estocástico (no determinístico).

Durante la época t se toma al azar un patrón x de P . El patrón x se compara con todos los vectores de pesos de las neuronas y se determina cuál es la neurona con vector más similar, que recibe el nombre de neurona ganadora o *BMU* (*Best Matching Unit*).

$$\text{dist}(x, w_*) = \min_{i=1 \dots n} (\text{dist}(x, w_i))$$

A la hora de determinar cuál es la neurona con vector de pesos más cercano a x pueden utilizarse diferentes variantes en la función de distancia. Por lo general, esa distancia depende fuertemente del problema al cual se aplique el SOM.

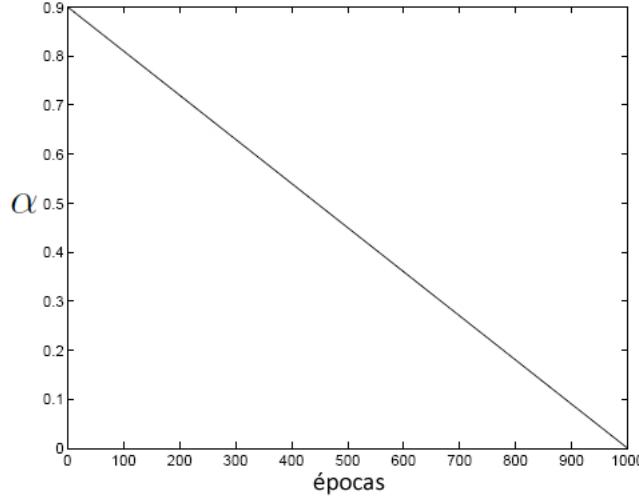


Figura 4.2: Tasa de aprendizaje lineal

Una vez obtenida la BMU, se procede a actualizar la red. La idea es que tanto la ganadora como neuronas cercanas a ella hagan que sus vectores de pesos se acerquen al patrón de entrada. Para esto se utilizarán dos funciones que deben ser decrecientes en el tiempo para garantizar la convergencia del algoritmo [8]. La primera es la tasa de aprendizaje ($\alpha(t)$), la cual define cuánto se actualizarán los pesos de las neuronas. La segunda es la función de vecindad ($h_{i,j}(t)$) que determina qué neuronas vecinas a la BMU se actualizarán y cómo se modificarán los pesos en función a la distancia a la BMU (cuanta más distancia, menos se actualizará).

Como dijimos anteriormente, la tasa de aprendizaje es una función decreciente con respecto al tiempo. Una de las más utilizadas es la función lineal (Fig. 4.2).

$$\alpha(t) = \alpha_{inicial} - t \left(\frac{\alpha_{inicial} - \alpha_{final}}{épocas} \right)$$

En cuanto a la función de vecindad, también es decreciente en el tiempo y además decreciente con respecto a la distancia entre la neurona ganadora y la que se desea actualizar. Típicamente, se usa la función *Gaussian*a, que tiene forma de campana con centro en la neurona ganadora (Fig. 4.3).

$$h_{i,j} = \exp \left(-\frac{\text{dist}^2(i,j)}{2\sigma^2(t)} \right)$$

$$\sigma(t) = \sigma_{inicial} - t \left(\frac{\sigma_{inicial} - \sigma_{final}}{épocas} \right)$$

Notar que para definir la función de vecindad se precisa establecer una función de distancia entre posiciones de neuronas (por lo general se utiliza la norma L2) y una función σ , que es la que determina el radio de neuronas que se actualizarán en ese paso y típicamente se usa una función lineal.

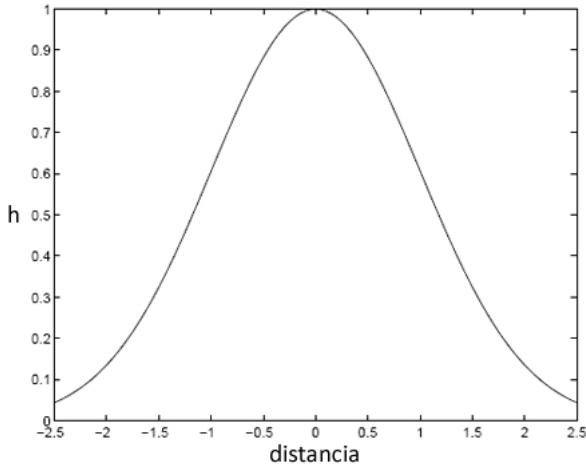


Figura 4.3: Función de vecindad Gaussiana

Luego, es necesario definir la regla de actualización de los pesos asociados a las neuronas en cada paso del algoritmo:

$$w_i := w_i + \alpha(t) h_{i,*}(t) (x - w_i)$$

En cuanto a las etapas de entrenamiento, se suelen distinguir dos: ordenamiento y convergencia. Durante la etapa de ordenamiento, se utiliza una tasa de aprendizaje alta y un radio que abarca gran parte de la red o incluso toda la red, disminuyendo hasta alcanzar pocas neuronas o solo la BMU. Luego, en la etapa de convergencia, se usan valores más bajos para la tasa de aprendizaje y el radio incluye a las vecinas inmediatas de la ganadora o únicamente a la neurona ganadora hasta terminar abarcando solamente a la BMU.

Rotulado

Una vez finalizada la etapa de entrenamiento, a cada neurona de la red es posible asignarle un rótulo. Existen diferentes maneras de rotular las neuronas y la forma de asignarles rótulos dependerá de la aplicación donde se utilice la red neuronal. Una de las alternativas es tomar los patrones mapeados a una neurona, quedarse con el más parecido al vector de pesos de la neurona y rotularla con algún dato de ese patrón.

4.3. Características principales del SOM

Una de las características más importantes que poseen los SOM es que *aproximan el espacio de entrada*. Este hecho se da a partir de que los pesos de las neuronas del mapa aproximan a los elementos del espacio de entrada. Es decir, un conjunto de N entradas es aproximado por los n vectores de pesos de las neuronas del mapa (típicamente $N \gg n$). Esta característica puede verse como una compresión del espacio de entrada.

Otra propiedad que poseen los SOM es la de *preservar el orden topológico*. El mapa hace que a elementos cercanos en el espacio de entrada les correspondan la misma neurona o neuronas cercanas. Al mismo tiempo, para elementos de la entrada alejados, se les asociarán

salidas alejadas.

Además de las anteriores cualidades, el SOM se caracteriza también por *respetar la densidad de los datos*. Es decir, regiones con mayor densidad de elementos en el espacio de entrada, en la grilla recibirán mayor cantidad de neuronas que regiones de menor densidad.

4.4. Variantes del SOM

Desde su aparición en la década del 80', los SOM han sido objeto de estudio y se han aplicado a una gran cantidad de trabajos en diferentes áreas. Hasta el día de hoy, se han desarrollado numerosas variantes intentando atacar diferentes aspectos del SOM.

Sin lugar a dudas, uno de los puntos que más interés ha acaparado es el que se relaciona con la escalabilidad de los SOM. Entrenar grandes redes, con grandes conjuntos de datos, puede requerir un gran costo computacional para el algoritmo de SOM tradicional. Este requerimiento ha sido el origen de diferentes variantes al SOM y en las siguientes secciones presentaremos algunas de las más importantes.

4.4.1. BSOM

Batch SOM (BSOM) es una variante del SOM tradicional desarrollada por Kohonen en 1990 [8] y su principal diferencia radica en el algoritmo de entrenamiento.

La versión *batch* consiste en modificar la forma en la cual se presentan las entradas durante cada época y la manera en la cual se actualiza la red. Dentro de una época, en vez de modificar la red por cada entrada, primero se calculan las neuronas ganadoras para todas las entradas y luego con toda esa información se actualiza la red. Una consecuencia directa de esta nueva forma de entrenamiento es que el algoritmo deja de ser estocástico, haciendo que sus resultados sean reproducibles.

La nueva regla de actualización se define de la siguiente manera:

$$w_i := \frac{\sum_{j=1}^N h_{i,*_j}(t)x_j}{\sum_{j=1}^N h_{i,*_j}(t)}$$

En la anterior fórmula, se denomina $*_j$ a la BMU del patrón x_j .

En términos de complejidad temporal, el algoritmo *online* y el *batch* tienen el mismo orden de complejidad asintótico, sin embargo, las constantes asociadas al *batch* son menores. Un análisis más detallado sobre este punto puede encontrarse en el Anexo A. La ventaja a favor del algoritmo *batch* se hace más visible cuanto mayor es la diferencia entre la cantidad de patrones de entrenamiento (N) y la cantidad de neuronas (n), siendo típicamente $N >> n$. Sobre el final de este capítulo se hará una comparación de los tiempos de cada una de estas variantes.

Como dijimos previamente, esta nueva forma de actualizar los pesos de las neuronas convierte al algoritmo en determinístico. Dependiendo del tipo de aplicación en el que se quiera

usar, el hecho de que los resultados sean reproducibles puede llegar a ser importante.

En la regla de actualización *batch* desaparece la tasa de aprendizaje. Este hecho hace que existan menos parámetros a la hora de entrenar la red, reduciendo la dificultad para fijar valores en la etapa de entrenamiento.

Por último, la herramienta *SOM Toolbox* [35], desarrollada por la Universidad de Helsinki, permite trabajar con ambas versiones del SOM pero recomienda trabajar con la *batch* dado que es más rápido el entrenamiento y los resultados son tan buenos o incluso mejores que la versión *online* [36] (de hecho, para entrenar un SOM, por defecto se utiliza la versión *batch*).

4.4.2. ParSOM

Parallel SOM (ParSOM) es una variante del SOM orientada a ambientes de ejecución en paralelo. Esta idea fue desarrollada por Tomsich, Rauber y Merkl y se publicó en el año 2000 [9].

Esta alternativa consiste en dejar intacto el algoritmo tradicional del SOM (versión *online*), pero permitiendo la ejecución de algunas tareas en paralelo. El principal objetivo es reducir el tiempo de búsqueda de la neurona ganadora y el tiempo de actualización de pesos de las neuronas una vez conseguida la ganadora. Para lograr esto se contará con diferentes nodos de procesamiento y a cada uno de ellos se le asignará una parte de la red (ver Fig. 4.4). Es importante destacar que al no modificar el algoritmo de entrenamiento, los resultados que valían para el SOM tradicional siguen valiendo para este enfoque.

En este esquema, un nodo trabajará como *master* y en cada época distribuirá una muestra para que cada nodo busque en su región de la red a la BMU. Cuando todos hayan calculado su neurona ganadora localmente, el nodo *master* establecerá la ganadora global y distribuirá este dato a los demás nodos para que actualicen su parte de la red, tal como indica el algoritmo original. Por cómo trabaja esta variante, se dice que es un algoritmo de *network partitioning*.

Es importante destacar que al reducir los tiempos de búsqueda de la BMU, se obtienen dos mejoras importantes. Por un lado, se optimiza el algoritmo de entrenamiento, reduciendo los tiempos de esta etapa. Por otro lado, aplicaciones que utilizan a la red entrenada y requieren buscar a la BMU, se benefician ya que también hacen uso de la búsqueda en paralelo.

Con la actual tendencia a ejecutar tareas en paralelo, *ParSOM* es sin lugar a dudas una idea prometedora, sobre todo en contextos donde se trabaja con gran cantidad de patrones de entrada.

4.4.3. Variante propuesta: ParBSOM

En este trabajo, proponemos una variante llamada *Parallel Batch SOM* (ParBSOM). La idea es combinar las características de ejecución en paralelo del ParSOM con las ventajas del BSOM. De esta forma será posible obtener una reducción en los tiempos por la ejecución en paralelo (heredada del ParSOM) y por la modificación en el algoritmo (tomada del BSOM). Al igual que en ParSOM, la ejecución en paralelo permite reducir los tiempos de entrenamiento y,

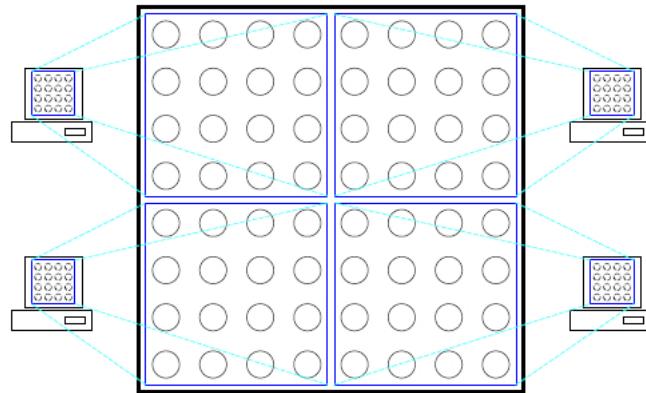


Figura 4.4: En ParSOM, la red es dividida en varias regiones y cada nodo de procesamiento trabaja exclusivamente con ella [1]

al mismo tiempo, beneficia a aplicaciones que requieren buscar la BMU en redes ya entrenadas.

Al igual que el ParSOM, el ParBSOM distribuirá la red entre varios nodos de procesamiento, permitiendo la ejecución en paralelo de la búsqueda de la ganadora y de la actualización de la red. Se encuadra dentro de los algoritmos de *network partitioning*. Un punto a destacar es que esta variante es una optimización de la versión *batch* del SOM, por lo que todos los resultados que valen para el BSOM siguen siendo válidos en este enfoque.

En los Algoritmos 5 y 6 se presentan los algoritmos de entrenamiento para BSOM y ParBSOM respectivamente (las diferencias están marcadas con color). Como puede observarse, al comienzo el ParBSOM distribuye las regiones de la red entre los nodos de procesamiento. Luego, se realizan en paralelo la búsqueda de la neurona ganadora y la actualización de cada región de la red.

Algoritmo 5 Algoritmo entrenamiento BSOM

Inicio

Para cada $t \leftarrow 1 \dots$ épocas

Para cada Patrón x de patrones

```
ganadora  $\leftarrow$  buscarGanadora(som, x)
guardar(listaGanadoras, <x, ganadora>)
```

Fin para

Para cada Neurona m de som

```
actualizarNeurona(m, listaGanadoras)
```

Fin para

Fin para

Fin

Algoritmo 6 Algoritmo entrenamiento ParBSOM

Inicio

$R_1, \dots, R_K \leftarrow \text{dividirEnRegiones}(\text{som}, K)$

Para cada $t \leftarrow 1 \dots \text{épocas}$

Para cada *Patrón* x de patrones

Hacer en paralelo $i \leftarrow 1 \dots K$

 ganador $_i \leftarrow \text{buscarGanadora}(R_i, x)$

Fin paralelo

 ganadora $\leftarrow \text{buscarGanadoraGlobal}(\text{ganadora}_1, \dots, \text{ganadora}_K)$

 guardar(listaGanadoras, $\langle x, \text{ganadora} \rangle$)

Fin para

Hacer en paralelo $i \leftarrow 1 \dots K$

Para cada *Neurona* m de R_i

 actualizarNeurona(m , listaGanadoras)

Fin para

Fin paralelo

Fin para

Fin

4.4.4. Otras variantes

TS-SOM

Tree Structured SOM es un SOM de estructura jerárquica y estática, que fue desarrollado por Koikkalainen y Oja en 1990 [10].

El TS-SOM posee diferentes niveles, siendo cada uno un SOM. Cada neurona del nivel superior tiene asociado un número fijo de neuronas en el nivel inferior (típicamente 4 hijos por cada neurona). El tamaño de cada SOM aumenta exponencialmente a medida que se baja a través de los niveles. El principal objetivo es reducir el tiempo de búsqueda de la neurona ganadora, pasando de un orden $O(n)$ a $O(\log n)$.

Para entrenar a un TS-SOM, se realiza el entrenamiento de cada nivel en forma separada. Se comienza entrenando el primer nivel, cuando el entrenamiento finaliza, se prosigue con el siguiente nivel y así sucesivamente. La característica más importante de esta forma de entrenamiento es que para buscar la neurona ganadora de un nivel i , no se busca en toda la red del nivel i , sino que se usa la estructura de árbol del TS-SOM. Más precisamente, se comienza en el primer nivel, donde se busca la ganadora en toda la capa. Luego, se pasa al siguiente nivel ($i + 1$) pero la búsqueda se limita a los hijos de la ganadora en el nivel i .

Un inconveniente que trae este enfoque es que dos entradas mapeadas a neuronas cercanas en un nivel superior, a medida que se baja en el árbol se irán alejando debido a que la búsqueda

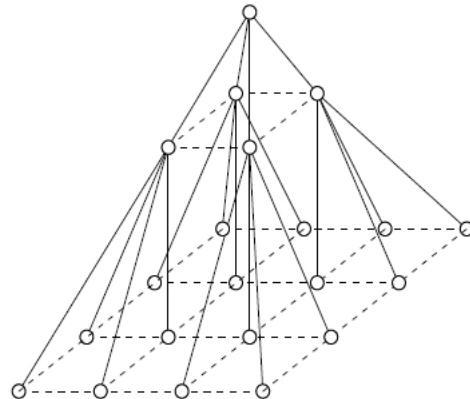


Figura 4.5: Estructura de un TS-SOM de 3 niveles [2]

va siendo más restringida. Para atacar este problema, se introduce el concepto de *wide-search*, ampliando la cantidad de neuronas que se consideran en los niveles inferiores. En el nivel $i+1$, en lugar de limitarse a las neuronas hijas del nivel i , se incluyen neuronas vecinas a las hijas, aumentando la cantidad de neuronas que se recorren.

En cuanto al espacio ocupado, la estructura de árbol que utiliza el TS-SOM requiere una mayor cantidad de recursos comparada con la que necesita una red en el SOM tradicional.

Por último, dado que el algoritmo del SOM no es exacto, los errores que se dan en capas superiores se propagarán a las inferiores, reduciendo significativamente la calidad del SOM definitivo [27]. Sin lugar a dudas, el hecho de que, a medida que se agregan niveles, se degrada la calidad del SOM, compromete seriamente la escalabilidad de este modelo.

GH-SOM

Growing Hierarchical SOM es un SOM de estructura jerárquica y dinámica, creado por Dittenbach, Rauber y Merkl en el año 2000 [3].

El GH-SOM consiste en varios niveles, donde cada nivel puede tener varios SOM independientes entre sí. El primer nivel está formado por un solo SOM. Cada neurona de este SOM puede tener asociado un nuevo SOM en el nivel inferior y esta idea se repite para los niveles subsiguientes. Se dice que es dinámico dado que puede crecer en dos sentidos: horizontal, donde un SOM de un nivel i puede aumentar el tamaño de su red; vertical, donde una neurona en el nivel i puede generar un nuevo SOM en el nivel $i+1$.

Su principal objetivo es proveer una estructura flexible, permitiendo que el tamaño se adapte a los datos de entrada y, al mismo tiempo, crear una estructura jerárquica para evitar trabajar con redes de enorme tamaño. Cabe destacar que la idea de adaptar la red a los datos de entrada no es nueva (previamente se desarrolló un SOM dinámico, Growing Grid [37]) pero el GH-SOM le agrega la estructura jerárquica.

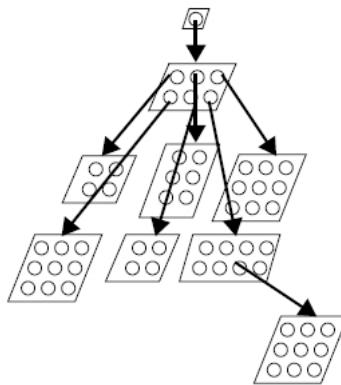


Figura 4.6: GH-SOM de 4 niveles [3]

En cuanto al entrenamiento, comienza con un SOM en el nivel 1 y primero se crece en forma horizontal hasta que se alcanza la calidad deseada, para luego dar lugar al crecimiento vertical. El entrenamiento vertical también se dará hasta que cierto umbral de calidad sea satisfecho.

Un problema que surge a partir de esta forma de entrenar la red es que se incorporan nuevas variables que deben ser definidas a la hora de entrenar al GH-SOM, cuando ya no es sencillo definir las existentes en el SOM tradicional. Además, se guía el entrenamiento en base a medidas de calidad como el error de cuantización, que son costosas de calcular y que no necesariamente dan lugar a redes de calidad. Por otro lado, el hecho de permitir un crecimiento desordenado de la estructura puede dificultar el uso de la red para tareas de visualización, una de las aplicaciones en las que puede utilizarse el SOM tradicional.

4.5. Experimentación

4.5.1. Descripción

El objetivo de estos experimentos es comparar el tiempo de entrenamiento de los siguientes tipos de SOM: *SOM Tradicional*, *ParSOM*, *BSOM* y *ParBSOM*. Se usarán conjuntos de entrenamiento (generados en forma aleatoria) con diferentes cantidades de datos y con muestras de diferentes dimensiones para evaluar en forma general el desempeño de los algoritmos.

Cabe aclarar que para los experimentos se utilizó una computadora con procesador *Intel Core 2 Duo* (2 Ghz) y 4 GB de memoria RAM. Los parámetros de entrenamiento detallados son descriptos en la Tabla 4.1.

Parámetros	Diferentes valores usados
Cantidad de muestras	1.000 5.000 10.000
Tamaño de muestra	250 500
Cantidad de neuronas de la red	100 500 1000 5000
Épocas	10
Función de vecindad	Gaussiana
Tasa de aprendizaje	Lineal
Tipo de entrenamiento	SOM ParSOM (2 threads) BSOM ParBSOM (2 threads)

Tabla 4.1: Parámetros usados para comparar tiempos de variantes de SOM

Muestras	Neuronas	SOM	ParSOM	BSOM	ParBSOM	Mejora BSOM vs SOM	Mejora ParBSOM vs ParSOM	Mejora ParSOM vs SOM	Mejora ParBSOM vs BSOM
[1.000 × 250]	100 500	3 s. 14 s.	2 s. 9 s.	1 s. 9 s.	1 s. 6 s.	67 % 36 %	50 % 33 %	33 % 36 %	0 % 33 %
[1.000 × 500]	100 500	5 s. 27 s.	4 s. 18 s.	2 s. 18 s.	2 s. 11 s.	60 % 33 %	50 % 39 %	20 % 33 %	0 % 39 %
[5.000 × 250]	100 500 1.000	13 s. 1,1 min. 2,37 min.	12 s. 44 s. 1,6 min.	5 s. 29 s. 1,1 min.	5 s. 19 s. 44 s.	62 % 56 % 52 %	58 % 57 % 54 %	8 % 33 % 33 %	0 % 34 % 35 %
[5.000 × 500]	100 500 1.000	25 s. 2,2 min. 4,4 min.	19 s. 1,5 min. 3,1 min.	10 s. 58 s. 2,2 min.	8 s. 38 s. 1,3 min.	60 % 55 % 50 %	58 % 58 % 57 %	24 % 31 % 31 %	20 % 34 % 40 %
[10.000 × 250]	100 500 1.000 5.000	26 s. 2,2 min. 4,7 min. 23,8 min.	23 s. 1,5 min. 3,2 min. 15,7 min.	10 s. 53 s. 2,0 min. 16,1 min.	10 s. 34 s. 1,2 min. 9,5 min.	62 % 60 % 58 % 32 %	57 % 63 % 61 % 40 %	12 % 32 % 32 % 34 %	0 % 36 % 37 % 41 %
[10.000 × 500]	100 500 1.000 5.000	49 s. 4,4 min. 9,1 min. 43,9 min.	35 s. 3,0 min. 6,2 min. 28,9 min.	20 s. 1,8 min. 4,0 min. 29,8 min.	15 s. 1,1 min. 2,4 min. 16,9 min.	59 % 59 % 56 % 32 %	57 % 63 % 61 % 41 %	29 % 31 % 32 % 34 %	25 % 38 % 40 % 43 %

Tabla 4.2: Comparación de tiempos para variantes del SOM (10 épocas de entrenamiento)

4.5.2. Resultados

En base a la Tabla 4.2¹ y a los Gráficos 4.7, 4.8, 4.9, 4.10, 4.11 y 4.12, lo primero que puede observarse es que efectivamente la versión *batch* del SOM requiere menos tiempo de entrenamiento que la versión tradicional. Como era de esperarse (según el análisis de costo temporal del Anexo A), cuanta mayor es la diferencia entre la cantidad datos y la cantidad

¹Para la experimentación, no se incluyen redes con más neuronas que cantidad de muestras porque no tienen sentido estos casos

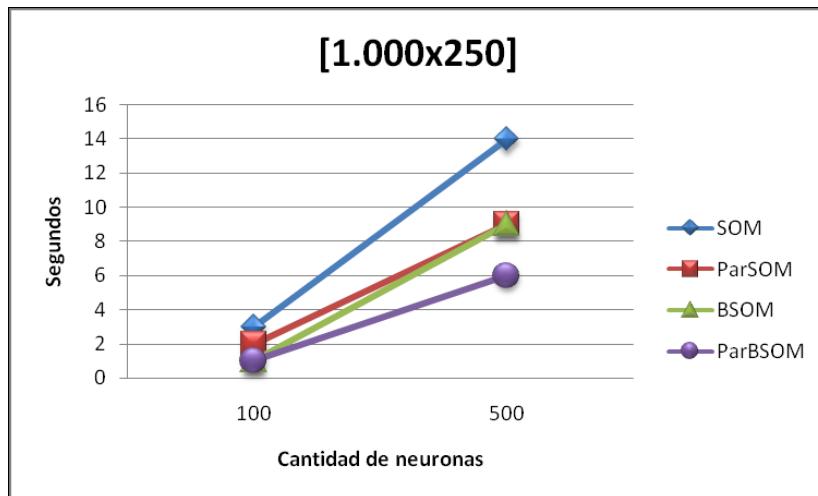


Figura 4.7: Tiempos para 1.000 muestras de dimensión 250

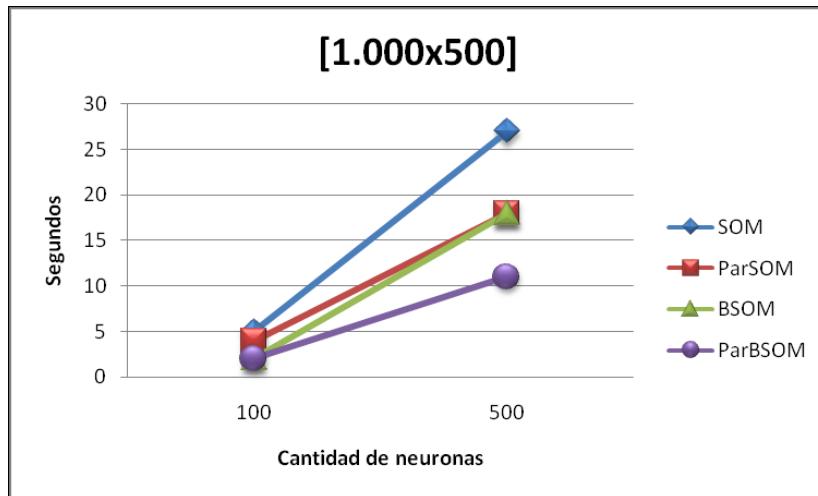


Figura 4.8: Tiempos para 1.000 muestras de dimensión 500

de neuronas, mejor es BSOM (valores superiores al 50 % de mejora).

En cuanto a las versiones paralelas (ParSOM y ParBSOM) puede notarse una mejora sustancial de los tiempos con respecto a las versiones secuenciales (no paralelas). A medida que aumenta la cantidad de neuronas, la ventaja que se obtiene es mayor. En algunos casos, se logra reducir más de un 40 % el tiempo de la versión secuencial. Estos resultados sugieren que la división del trabajo entre dos nodos de procesamiento produce una mejora importante con respecto a utilizar uno solo. Además, el hecho de que aumentando la cantidad de neuronas se mejoren todavía más los tiempos lo vuelve interesante por su escalabilidad.

Por último, si comparamos las versiones paralelas entre sí, la variante ParBSOM es la que tiene mejor desempeño, dado que combina al algoritmo *batch* con la parallelización del trabajo. En la mayoría de los casos, las mejoras llegan a superar el 50 %.

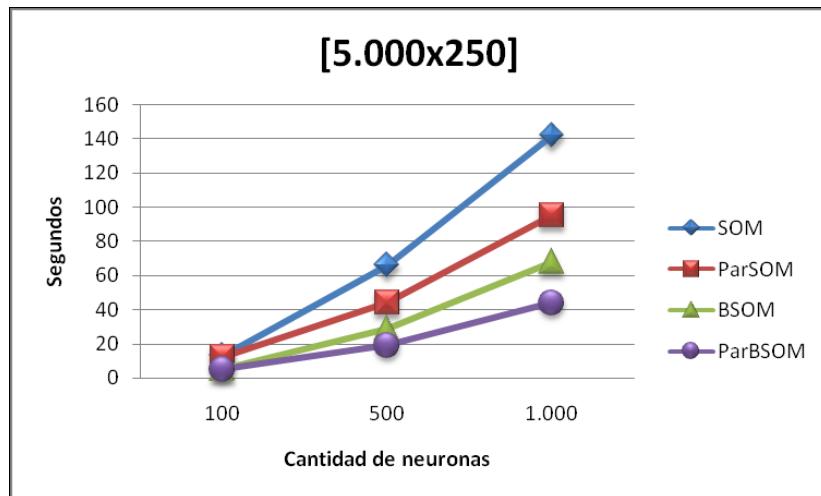


Figura 4.9: Tiempos para 5.000 muestras de dimensión 250

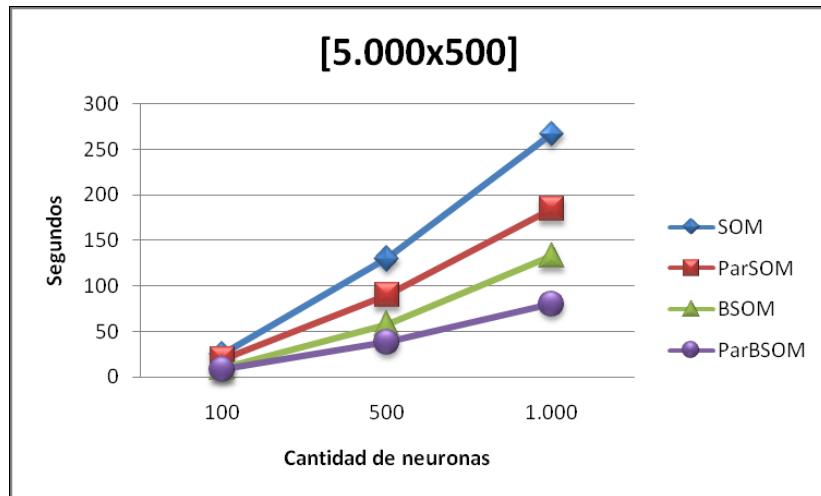


Figura 4.10: Tiempos para 5.000 muestras de dimensión 500

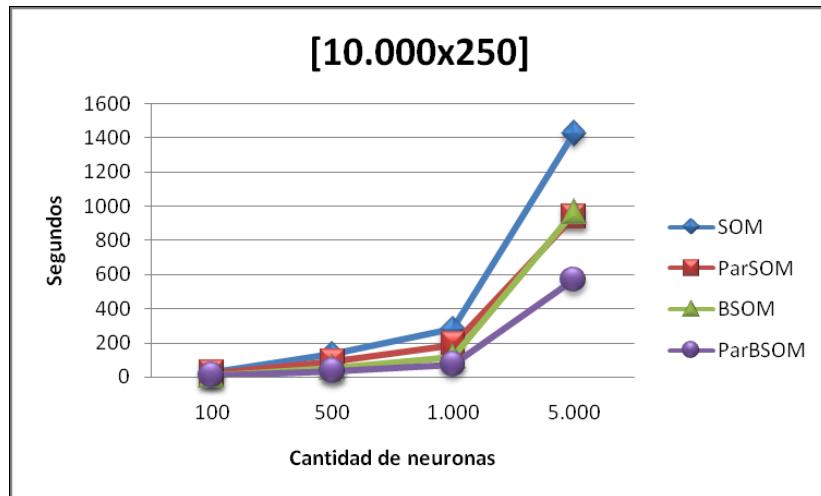


Figura 4.11: Tiempos para 10.000 muestras de dimensión 250

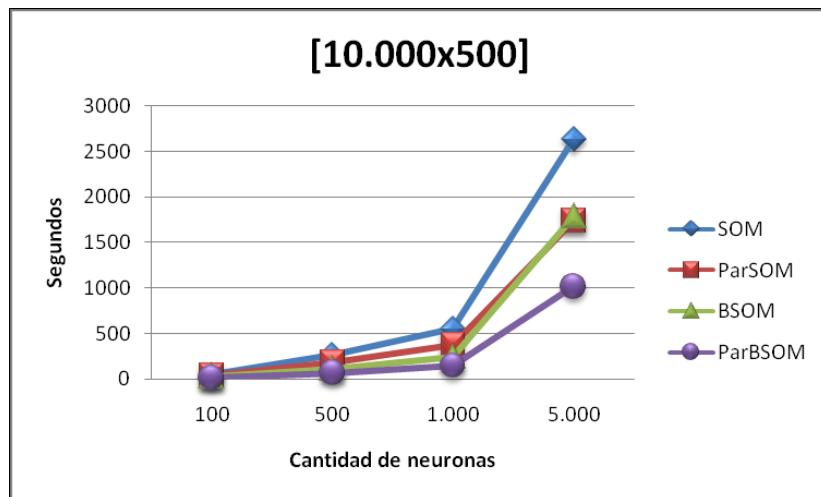


Figura 4.12: Tiempos para 10.000 muestras de dimensión 500

Capítulo 5

Indexación CBIR

La forma en la que los descriptores de las imágenes son almacenados determina gran parte de la eficiencia de un sistema de CBIR. A diferencia de TBIR, CBIR es un área relativamente nueva que aún presenta diversos desafíos y, sin lugar a dudas, el encontrar una estructura de índice adecuada es uno de los más interesantes. Los SOM se presentan como una alternativa viable, dado que son capaces de trabajar con grandes bases de datos de imágenes y permiten indexar descriptores de alta dimensionalidad. En este capítulo estudiaremos cómo los SOM pueden aplicarse a la indexación en CBIR.

5.1. Introducción

Dentro de un sistema de CBIR, la forma de indexar los descriptores generados para cada imagen no es una tarea trivial. En su forma más sencilla, una base de datos de imágenes es simplemente una colección de imágenes. Sin embargo, a medida que crece el tamaño de las bases de datos, se vuelve más notoria la necesidad de encontrar estructuras eficientes (índices) que permitan un acceso rápido a la información. Contrariamente a lo que sucede en TBIR donde existe un consenso acerca de los tipos de índices a utilizar (índices invertidos), en CBIR conviven varias estructuras pero ninguna de ellas ha logrado afirmarse definitivamente.

Para tratar de definir un índice adecuado, antes deben comprenderse las necesidades y características de los sistemas de CBIR. Por un lado, debe tenerse en cuenta que la indexación (al igual que la extracción de características de las imágenes) se realiza en forma *offline*, durante la construcción del sistema. Durante el funcionamiento *online* de la aplicación, sólo se procesan las consultas de los usuarios. Por este motivo, a la hora de elegir un índice se privilegia su eficiencia en la recuperación sobre el tiempo que demora en construirse. Por otro lado, los descriptores visuales se caracterizan por tener una alta dimensionalidad (aproximadamente, mayor a 100) [4], por lo que el índice debe ser capaz de manejar datos de este tamaño o superiores.

Entre las opciones disponibles para CBIR, el índice más sencillo consiste en almacenar los descriptores sin ningún orden especial y luego realizar una búsqueda lineal sobre las imágenes (*Fuerza Bruta*). Sin embargo, esta variante puede resultar imprácticable para grandes bases de datos. Una alternativa son los índices con estructura de árbol como KD-trees [6], R-trees [6], entre otras. Si bien estas variantes proveen una búsqueda de orden logarítmico

con respecto al tamaño de la base de imágenes, no escalan para dimensiones mayores a 20 [4]. Además, la mayoría asume que la distancia entre descriptores es la Euclídea, a pesar de que en muchos casos esta distancia no sea la más conveniente para simular la percepción humana [4].

Los *Self-Organizing Maps* (SOM) surgen como una posible estructura de índice para CBIR. Se caracterizan por ser capaces de trabajar con descriptores de alta dimensionalidad (escalan *linealmente*), por realizar una compresión del espacio de entrada soportando bases de datos de gran tamaño y, además, por brindar una alta flexibilidad permitiendo trabajar con diferentes nociones de distancia entre descriptores. En particular, estudiaremos la variante ParBSOM, que se destaca por ser más fácil y rápida de entrenar.

5.2. Construcción del índice

La construcción del índice consistirá básicamente en entrenar una red siguiendo los pasos del algoritmo del ParBSOM. Durante este proceso se utilizará como conjunto de entrenamiento únicamente a los descriptores de las imágenes de la base de datos.

Para comenzar a trabajar, se deberá definir el tamaño de la red, es decir, la cantidad de neuronas (n). Si la base de datos de imágenes consta de N imágenes, se buscará definir un valor n tal que $N \gg n$.

Cuando se define el tipo de descriptor a utilizar, debe elegirse también la función de distancia entre descriptores de imágenes. Esta función es la que se usará en la red como función de distancia entre pesos de las neuronas. Tal como mencionamos previamente, los SOM brindan una gran flexibilidad permitiendo utilizar diferentes funciones dependiendo del descriptor utilizado.

Una vez finalizada la etapa de entrenamiento, la red se adaptará a las entradas y agrupará imágenes similares, haciendo que imágenes parecidas sean mapeadas a la misma neurona o a una cercana, hecho que será utilizado durante la etapa de recuperación.

Por último, se llevará a cabo el rotulado de la red. Para cada descriptor se buscará su correspondiente BMU y la imagen de ese descriptor será asignada a esa neurona. Una vez finalizado este proceso, se designará como representante de la neurona a la imagen cuyo descriptor sea más parecido al vector de pesos de la neurona.

5.3. Recuperación a través del índice

Para recuperar imágenes a partir de una imagen usada como consulta, se procede de la siguiente manera (Fig. 5.1):

1. Se calcula el descriptor de la imagen.
2. Se busca en la red la BMU del descriptor.
3. Se obtienen las imágenes mapeadas a la BMU, ordenadas en base a la similitud con el descriptor de la consulta.

4. Se calcula el *score* (descripto en la Sección 3.7) de cada resultado y se devuelven aquellas que superen el umbral fijado por el sistema.

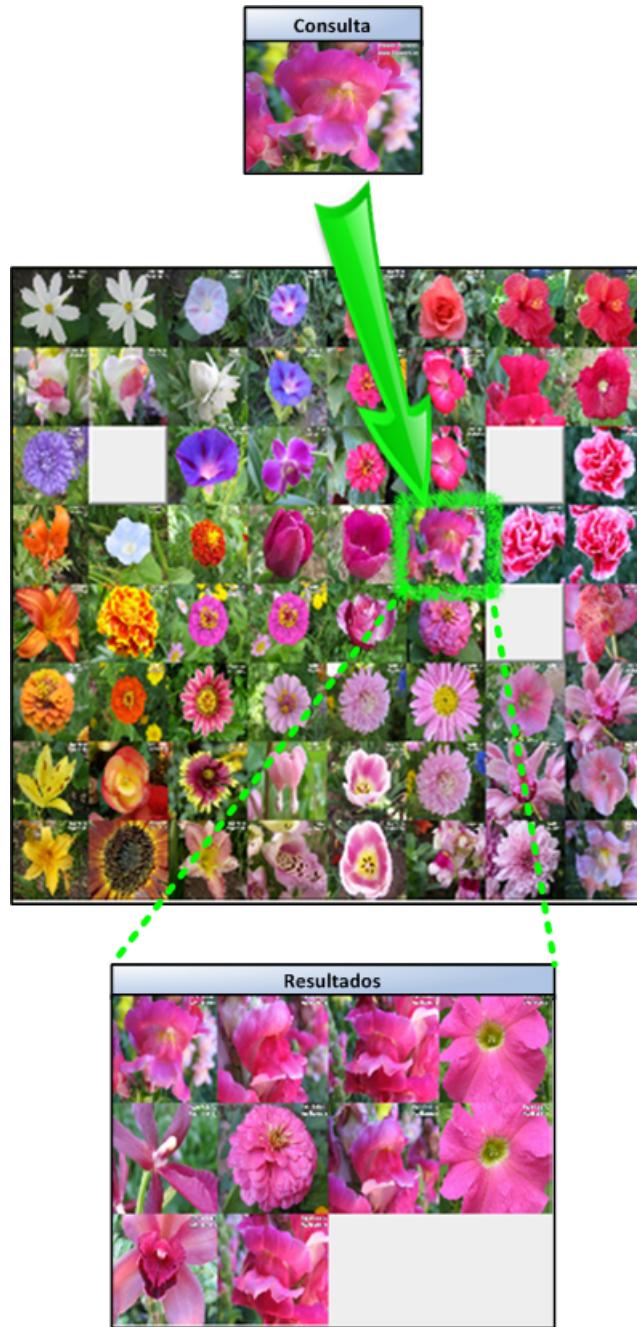


Figura 5.1: Recuperación de imágenes a través de un SOM

Una consecuencia importante es que al preservar el orden topológico de la entrada, es posible navegar la red buscando resultados similares (proceso conocido como *browsing*), ya que alrededor de la BMU de la consulta se podrán encontrar imágenes con características similares.

5.4. Experimentación

5.4.1. Descripción

Los experimentos presentados en esta sección buscarán comparar la calidad de los mapas generados por las variantes ParSOM y ParBSOM contra la variante de Fuerza Bruta, donde se realiza una búsqueda secuencial sobre las imágenes de la base de datos. Es importante recordar que esta comparación también será válida para las versiones SOM y BSOM, pues la única diferencia es que el entrenamiento paralelo es considerablemente más rápido.

Además de comparar la calidad, se medirá el tiempo que requieren la variante de Fuerza Bruta y la de los SOM en recuperar una imagen de la base (aquí no se distinguirá entre la versión *online* y la *batch* porque se diferencian en el entrenamiento pero no en la forma de recuperar las imágenes). Para medir los tiempos se utilizó una computadora con procesador *Intel Core 2 Duo* (2 Ghz) y 4 GB de memoria RAM.

Se trabajará con las bases de imágenes presentadas en la sección 2.6.1: *ZuBud*, *UCID* y *UK Bench* y se usarán las métricas *MAP*, *Precision*, *Recall* y *F-Measure* (descriptas en la sección 2.6.2).

Las imágenes serán descriptas a través de histogramas de color HSV de 512 *bins* (32x4x4) y se los comparará a través de la norma L1. Además, se utilizará un umbral de 0.7 (utilizado para filtrar imágenes irrelevantes para la consulta), con el que empíricamente se obtuvo mejor desempeño global en la sección 3.9.

En cuanto al entrenamiento de los mapas, se realizaron dos etapas. La primera etapa es la de ordenamiento, la cual usa un radio que comienza abarcando toda la red hasta alcanzar a las vecinas inmediatas de la BMU y una tasa de aprendizaje alta (recordar que esto último sólo aplica para la versión *online*). La siguiente etapa es la de convergencia, donde se utiliza un radio que va desde las vecinas inmediatas de la BMU hasta solamente la BMU y una tasa de aprendizaje baja. Además, para cada etapa se usaron como máximo 1000 épocas de entrenamiento.

Por último, el tamaño de los mapas es de 100 neuronas para la ZuBuD, 150 para la UCID y 1000 para la UK Bench.

5.4.2. Resultados

Bases de Imágenes	Fuerza Bruta				ParBSOM				ParSOM			
	MAP	Prec	Recall	F	MAP	Prec	Recall	F	MAP	Prec	Recall	F
ZuBuD	68,13	75,49	69,22	66,98	63,30	78,05	63,83	66,67	62,78	78,41	63,30	66,23
UCID	18,50	26,84	19,22	20,38	16,38	24,73	17,36	18,73	16,38	24,55	17,10	18,16
UK Bench	49,06	55,58	53,63	46,66	42,17	52,27	44,50	42,43	41,98	51,79	44,51	42,02

Tabla 5.1: Comparación de la calidad de recuperación de las técnicas de indexado

Como puede observarse en las Tablas 5.1 y 5.2 y en la Fig. 5.2, los enfoques que utilizan SOM presentan una pequeña pérdida de calidad, tal como era de esperarse por no ser algorit-

Bases de Imágenes	Pérdida ParBSOM vs Fuerza Bruta	Pérdida ParSOM vs Fuerza Bruta	Pérdida ParSOM vs ParBSOM
ZuBuD	0,46 %	1,12 %	0,66 %
UCID	8,1 %	10,89 %	3,04 %
UK Bench	9,07 %	9,94 %	0,97 %

Tabla 5.2: Pérdida con respecto a F-Measure de las técnicas de indexado

Bases de Imágenes	Tiempo Fuerza Bruta	Tiempo SOM	Mejora SOM vs Fuerza Bruta
ZuBuD	3,43 ms.	0,27 ms.	92 %
UCID	4,58 ms.	0,32 ms.	93 %
UK Bench	40,63 ms.	1,68 ms.	96 %

Tabla 5.3: Comparación del tiempo requerido por las técnicas de indexado para recuperar una imagen

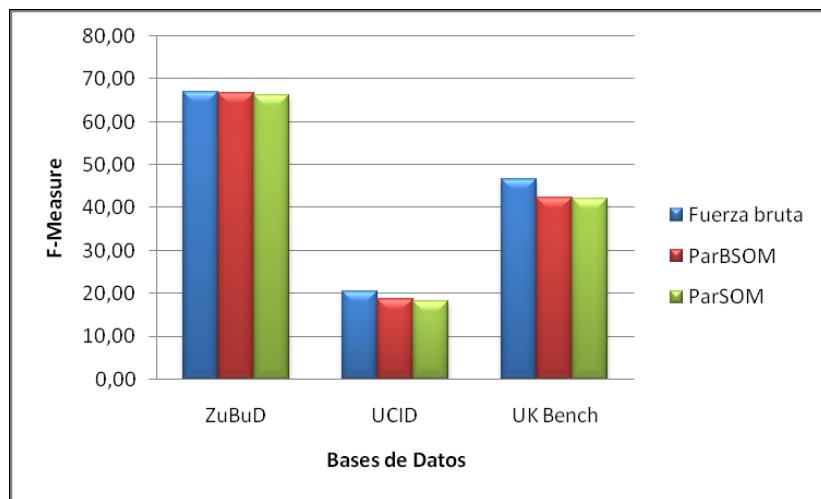


Figura 5.2: Gráfico de F-Measure para las bases de datos y las técnicas de indexación

mos exactos como el de Fuerza Bruta. Para la base de datos ZuBuD, las pérdidas son cercanas al 1%, mientras que en la UCID y la UK Bench, se mantienen menores al 10%.

Si comparamos la calidad de las versiones ParSOM y ParBSOM, se puede observar que la calidad de mapas generados es muy similar, no habiendo una clara ventaja de un enfoque sobre el otro (a diferencia de lo que sucedía en los tiempos de entrenamiento donde la versión *batch* era considerablemente más rápida).

En cuanto a los tiempos de recuperación (Tabla 5.3 y Fig. 5.3), la versión que utiliza como índice al SOM reduce en forma drástica los tiempos del enfoque de Fuerza Bruta. Las mejoras son superiores al 90%, siendo más notorias a medida que la base de datos es de mayor tamaño, donde la Fuerza Bruta puede resultar impracticable.

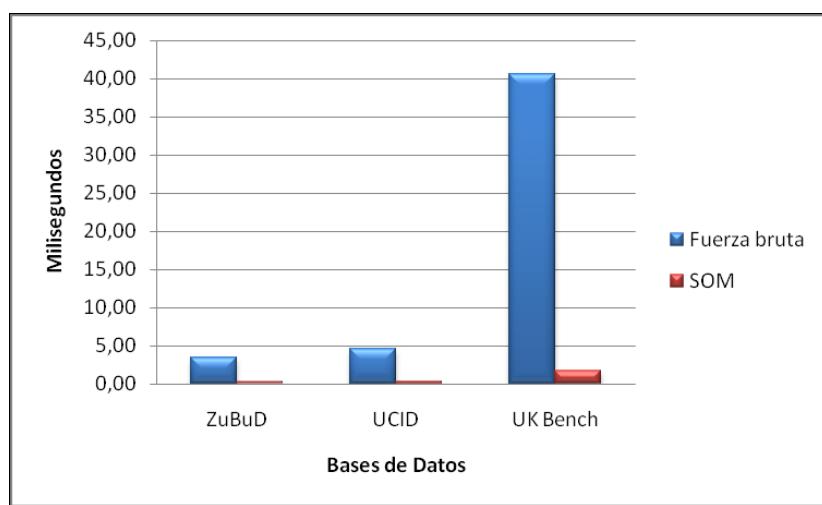


Figura 5.3: Tiempo requerido para recuperar una imagen para Fuerza Bruta y para un SOM en las bases de datos

Capítulo 6

Recuperación Híbrida

La recuperación híbrida consiste en combinar los enfoques de TBIR y CBIR, intentando reducir los problemas de cada uno por separado y, al mismo tiempo, potenciar las virtudes de ambos. Durante los últimos años se ha comenzado a estudiar esta variante y a lo largo de este capítulo nos dedicaremos a analizarla en profundidad.

6.1. Introducción

Desde su aparición a fines de los años 70¹, el área de VIR ha sido analizada desde diferentes enfoques. Como primera alternativa, surgió utilizar una descripción textual para representar a las imágenes (enfoque conocido como TBIR). Más adelante, se intentó representar a las imágenes exclusivamente a través de su contenido visual (CBIR). Sin embargo, tanto la recuperación basada en texto como la basada en contenido poseen sus desventajas. En TBIR, los textos que describen a las imágenes son claramente subjetivos y pueden ser vagos e imprecisos. A su vez, como CBIR utiliza características de bajo nivel de la imagen, es posible que se consideren parecidas a imágenes semánticamente diferentes. Con la idea de reducir las desventajas y potenciar las virtudes de cada enfoque, en los últimos años surgió la idea de utilizar sistemas híbridos, combinando la descripción a través de texto y de las características visuales.

El hecho de que dentro de un sistema de VIR se describan las imágenes a partir de su contenido textual y visual, plantea diversas alternativas a la hora de decidir cómo combinar los resultados de TBIR y CBIR. Sin lugar a dudas, éste es un nuevo desafío que se plantea y en las secciones siguientes será analizado en detalle.

6.2. Estructura de un Sistema Híbrido

Un sistema híbrido se caracteriza por combinar la descripción textual y visual para representar a las imágenes (Fig. 6.1). Internamente, es posible distinguir dos módulos: el que se encarga de trabajar con el texto y el que maneja el contenido visual.

Durante la construcción de los índices (etapa *offline*), cada módulo trabaja en forma independiente. Primero, se procesan los datos de entrada (texto o imágenes, según corresponda) y luego, se construyen las estructuras de índices.

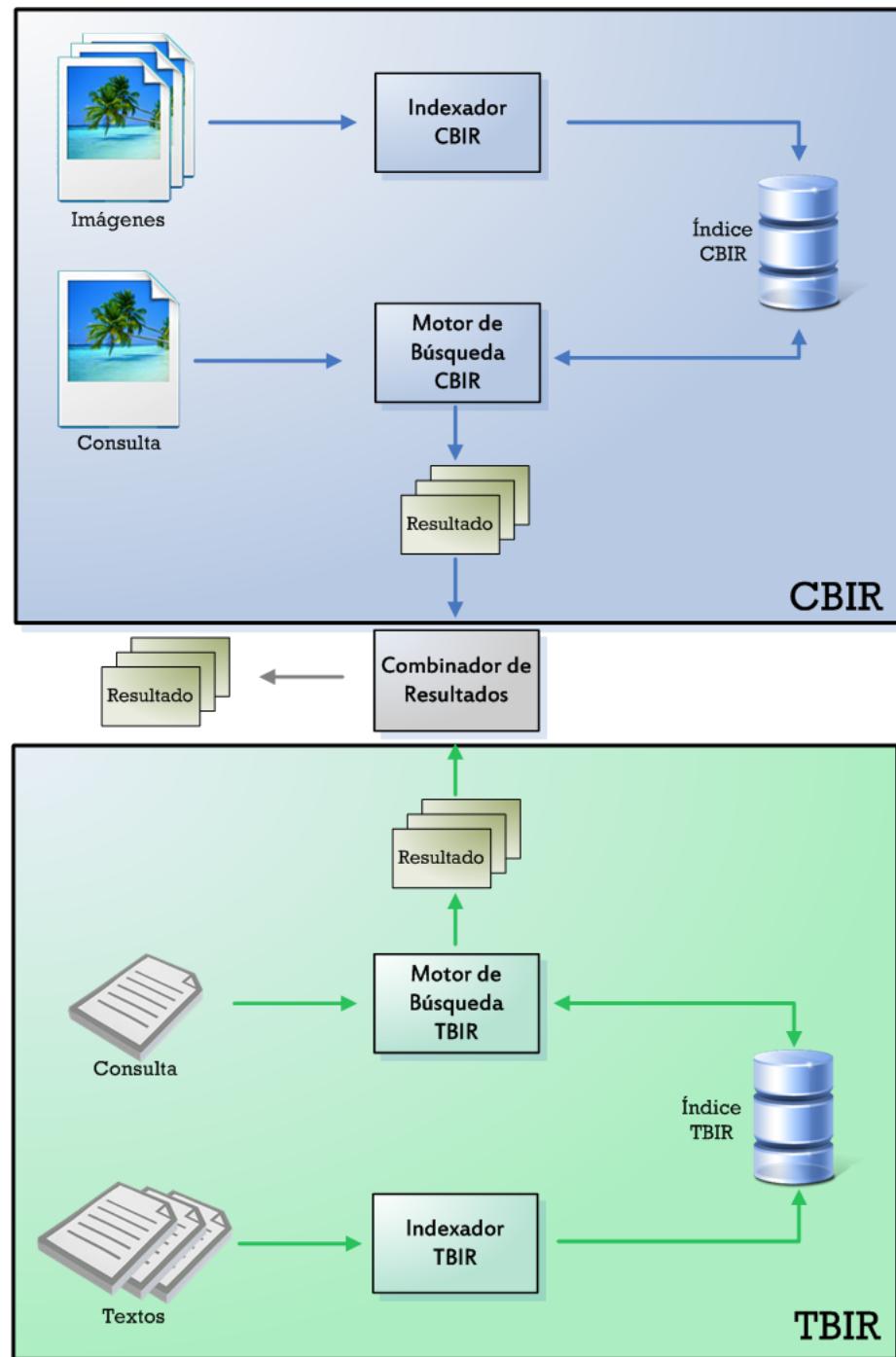


Figura 6.1: Estructura de un sistema de VIR siguiendo el enfoque híbrido

Para la recuperación de imágenes (etapa *online*), es posible formular consultas utilizando tanto texto como imágenes. Formalmente, una consulta puede notarse como $Q = (Q_T, Q_I)$ donde Q_T representa al texto que describe a la consulta y Q_I a la imagen de la consulta. Si $Q_I = \text{NULL}$, la consulta se procesa como TBIR; en cambio, si $Q_T = \text{NULL}$, es una consulta de CBIR. Ahora, si el usuario provee texto e imagen, cada módulo genera su propia lista de resultados y posteriormente se realiza una combinación de los resultados de TBIR y CBIR, que es la lista que finalmente se devolverá al usuario. Esta forma de combinar los resultados provenientes de texto y contenido visual se conoce en la literatura como *late fusion* y existen diversas estrategias para llevar esto a cabo.

6.3. Estrategias de combinación de resultados de TBIR y CBIR

En los sistemas híbridos, definir cómo se combinarán los resultados generados a partir de la búsqueda por contenido textual y visual es de vital importancia. El objetivo es complementar ambos enfoques, reduciendo los efectos negativos de cada uno y, al mismo tiempo, mejorando los resultados obtenidos de cada variante por separado.

A continuación presentaremos algunas de las estrategias utilizadas recientemente para combinar resultados textuales y visuales [4]. Nos referiremos a la lista de resultados de TBIR como R_T y a la de CBIR como R_I .

6.3.1. Votación

En este esquema se modela al problema como una votación, donde hay jurados y candidatos. Los candidatos son las imágenes de $R_T \cup R_I$ y los jurados son el sistema de TBIR y el de CBIR. Cada jurado arma su lista de candidatos y le asigna un puntaje a cada uno. Suponiendo que la lista de candidatos de un jurado es de tamaño n , se asignan puntajes de la siguiente forma: el primer elemento obtiene n puntos, el segundo $n - 1$ y así sucesivamente. La lista final de resultados se obtiene ordenando el puntaje total de cada candidato de mayor a menor.

Esta estrategia favorece a imágenes que tienen ranking alto en ambas listas y le da la misma importancia a imágenes provenientes de TBIR y CBIR. Esto último puede verse como una desventaja, ya que en los resultados de CBIR suelen aparecer imágenes visualmente similares pero semánticamente irrelevantes para la consulta. Si bien existen variantes sobre esta estrategia que permiten darle más peso a los votos de texto o de imagen, puede no ser sencillo definir estos valores.

6.3.2. Refinamiento

La estrategia de refinamiento consiste en refinar los resultados basados en texto utilizando los resultados basados en contenido visual. Es decir, R_T es reordenado a partir de R_I . La lista de resultados final contendrá a las mismas imágenes que provienen de TBIR, pero el ordenamiento podrá ser diferente dependiendo de los resultados de CBIR.

La idea de esta estrategia es quedarse únicamente con los resultados que son semánticamente significativos para la consulta (o sea, los resultados de TBIR) y ordenarlos en base a

su similitud visual. De esta forma, imágenes que son parecidas visualmente pero que semánticamente son diferentes serán descartadas de la lista final.

Dado que actualmente el área de TBIR se encuentra más desarrollada que la de CBIR, otorgarle mayor importancia a los resultados basados en texto puede resultar beneficioso y ésta será la estrategia con la cual trabajaremos.

6.4. Sistema propuesto: *Envision*

Con la idea de facilitar la experimentación de diferentes variantes asociadas a VIR, desarrollamos un sistema al que denominamos *Envision*. La implementación se realizó en lenguaje *Java*, lo que lo convierte en un motor multiplataforma.

Dado que fue concebido con el objetivo de permitir introducir fácilmente variantes para VIR, la implementación se caracteriza por tener un esquema altamente *modular*. Por ejemplo, incluir nuevos descriptores de imágenes, tipos de índice para texto o imágenes se convierten en tareas sencillas.

Para la parte de CBIR se utilizan estructuras desarrolladas íntegramente por nosotros. En cuanto a TBIR, actualmente se hace uso del motor de texto *Terrier*[38, 39], desarrollado por la Universidad de Glasgow.

A continuación describimos las características generales de este sistema y en las Fig. 6.2 y 6.3 presentamos capturas del funcionamiento típico del motor.

CBIR	
Extracción de características	Histograma de color HSV 512 bins (32x4x4) Función de distancia L1 Uso de <i>umbral</i> para filtrar imágenes irrelevantes
Indexación	ParBSOM
TBIR	
Extracción de características	Normalización de caracteres Stopwords Stemming (algoritmo de Porter)
Indexación	Índices Invertidos (modelo tf-idf)
CBIR + TBIR	
Recuperación	Automática (sin <i>relevance feedback</i>) Estrategia de <i>refinamiento</i> para combinar resultados

6.5. Experimentación

6.5.1. Descripción

El objetivo de los siguientes experimentos será analizar los efectos que produce la combinación de resultados textuales y visuales. Se utilizará la base de datos de la *ImageCLEFphoto 2007* (descripta en sección 2.6.1, que contiene imágenes y una descripción textual asociada a cada una). La evaluación se realizará usando las métricas *MAP*, *Precision*, *Recall*, *F-Measure*, *Precision at 10* (*P(10)*) y *Precision at 20* (*P(20)*) (descriptas en la sección 2.6.2).

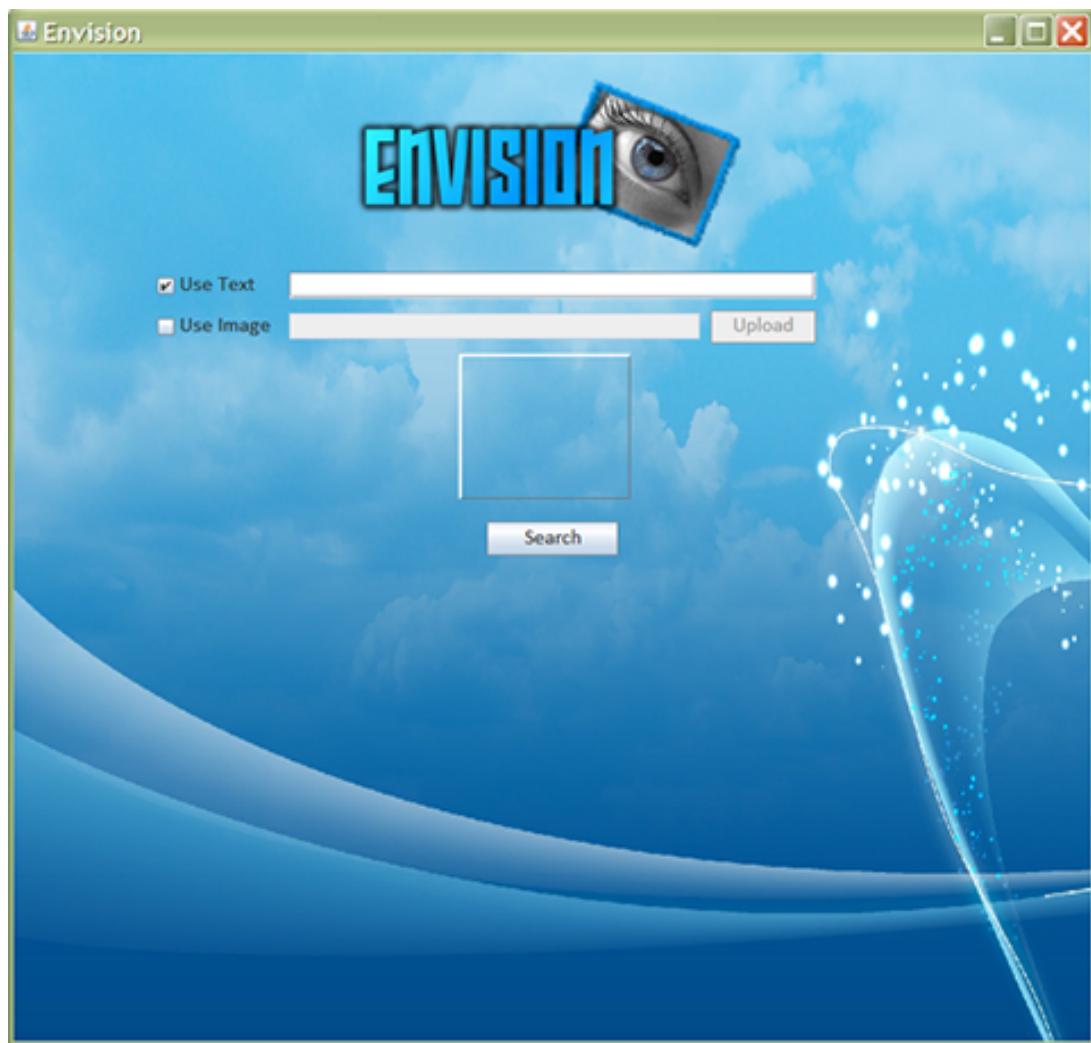


Figura 6.2: *Envision*: La consulta puede consistir en texto, una imagen o ambas alternativas

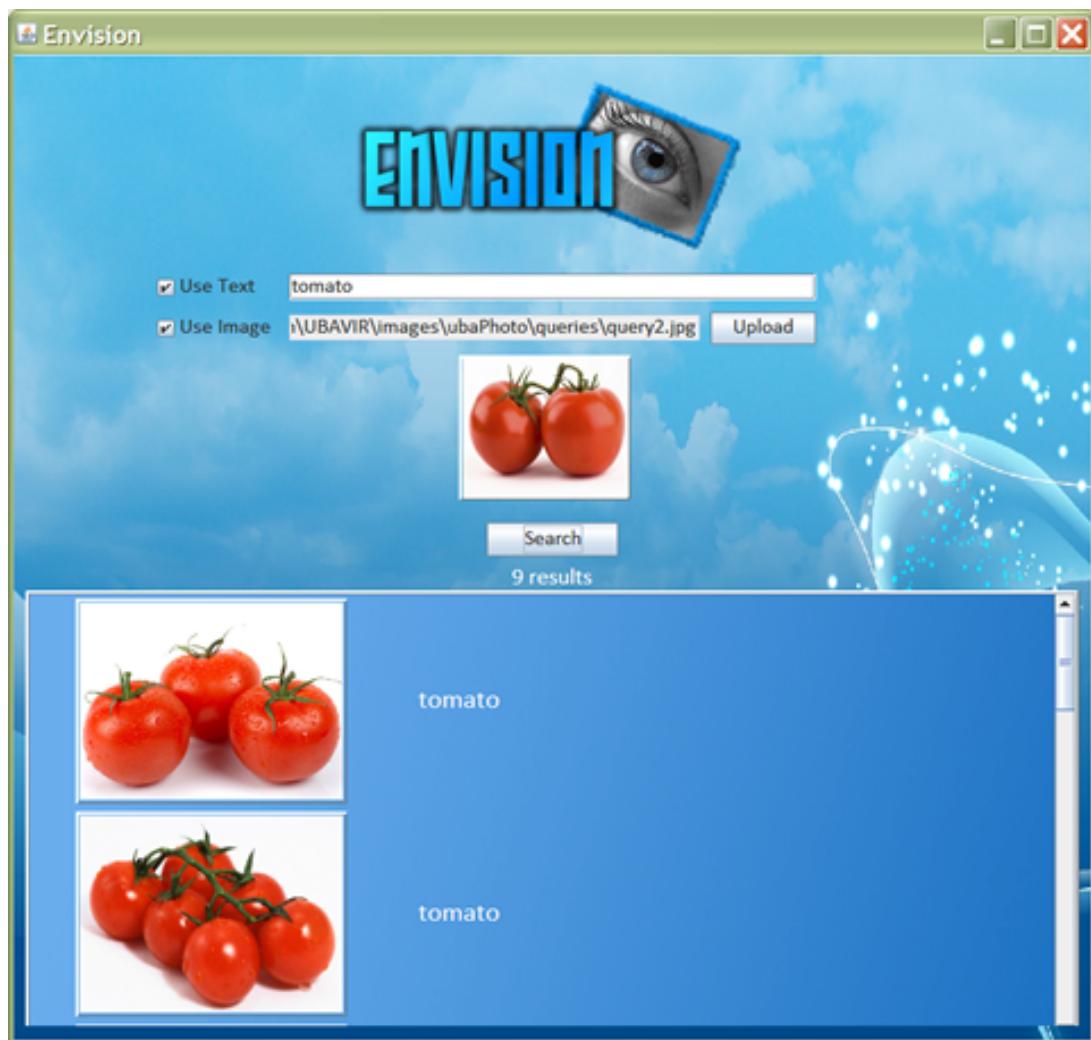


Figura 6.3: *Envision*: A partir de texto y una imagen, se recuperan imágenes relevantes para la consulta

En cuanto al contenido visual, como descriptor se utilizará el histograma de color HSV de 512 *bins* (32x4x4) y se lo comparará a través de la norma L1. Además, se utilizará un umbral de 0.6 ya que experimentalmente resultó ser el más adecuado para esta base (se fijó siguiendo el mismo criterio usado en la Sección 3.9). Para la indexación de los descriptores se empleará un ParBSOM de 1500 unidades.

Para la descripción textual se usarán índices invertidos, con normalización de caracteres, uso de *stopwords* y *stemming* (algoritmo de Porter [25]).

La combinación de resultados de TBIR y CBIR se realiza siguiendo la estrategia de refinamiento.

6.5.2. Resultados

Tipo	MAP	Prec.	Recall	F	P(10)	P(20)
CBIR	4,41	33,69	5,56	8,50	20,17	12,83
TBIR	14,94	5,35	49,27	8,26	22,33	18,33
TBIR + CBIR	16,59	5,35	49,27	8,26	27,83	22,08
Mejora TBIR + CBIR vs. TBIR	9,95 %	0 %	0 %	0 %	19,76 %	16,98 %

Tabla 6.1: Comparación de tipos de recuperación en la base ImageCLEFphoto 2007

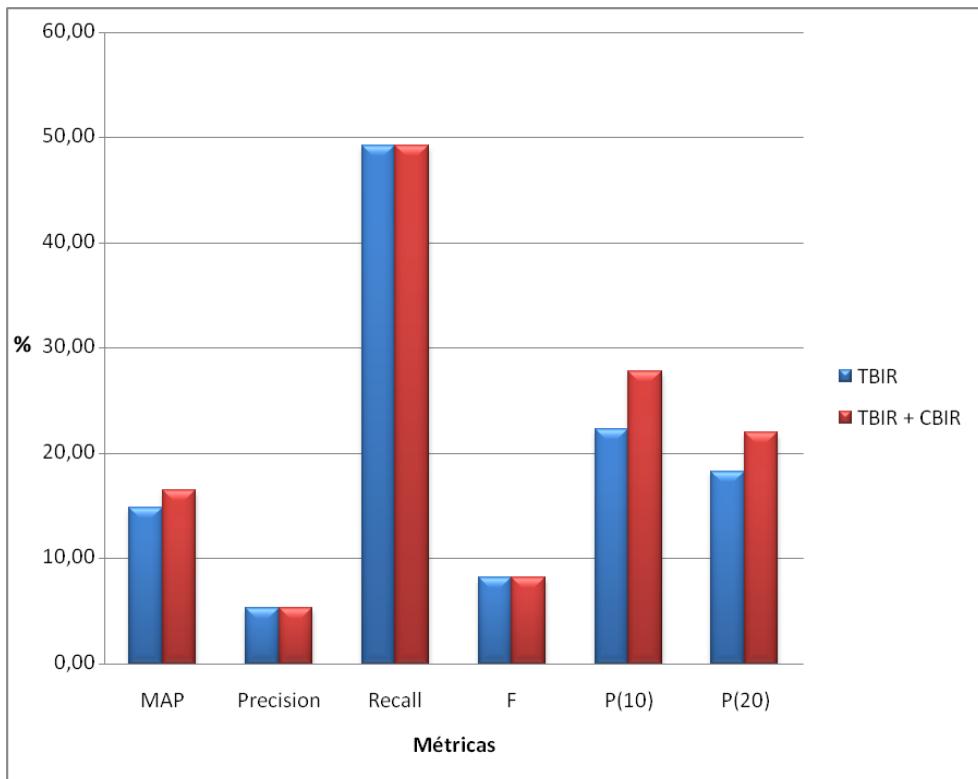


Figura 6.4: TBIR vs. TBIR+CBIR para diferentes métricas en la base ImageCLEFphoto 2007

En la Tabla 6.1 y en la Fig. 6.4 se realiza una comparación entre los distintos tipos de recuperación. Comparando CBIR con TBIR, se puede apreciar que CBIR obtiene una mayor precisión y un menor recall que el segundo, mientras que en TBIR se da la situación inversa, dando lugar a similares valores de F-Measure. En el enfoque que combina TBIR con CBIR (usando la variante de *refinamiento*), los valores de precisión, recall y F-Measure se mantienen iguales a los de TBIR, tal como era de esperarse (no cambian las imágenes devueltas pero sí puede cambiar el ranking de las mismas). A diferencia de estas métricas que no son sensibles a cambios en los rankings, el MAP consigue aumentar casi un 10 % dado que imágenes relevantes se ubican en posiciones más altas del ranking. Además, al considerar la precisión con respecto a los primeros 10 y 20 resultados, se observa una gran mejora con respecto a TBIR, obteniendo una ganancia de casi 20 %.

Capítulo 7

Conclusiones y trabajos futuros

7.1. Conclusiones

A lo largo de este trabajo, se realiza un profundo estudio acerca del estado del arte del área de *Visual Information Retrieval* (VIR) y se proponen técnicas orientadas a ser aplicadas en la recuperación híbrida, la cual combina la descripción basada en texto y características visuales. Este enfoque ha comenzado a ser estudiado durante los últimos años, surgiendo como propuesta ante las desventajas de las técnicas que usan texto o contenido visual por separado.

Debido a que la recuperación basada en texto (TBIR) se encuentra considerablemente más madura que la basada en contenido visual (CBIR), nuestro trabajo se centró en avanzar sobre aspectos aún no resueltos en el área de CBIR. Partiendo de un descriptor simple y ampliamente usado en el área como los histogramas de color, nuestro principal propósito fue estudiar la utilización de los mapas auto-organizados de Kohonen (SOM) como índices para CBIR. Nuestro objetivo final consistió en analizar cómo estos resultados podían aplicarse en el contexto de recuperación híbrida, apuntando a seguir avanzando en esta reciente línea de investigación.

Para llevar a cabo nuestro trabajo, decidimos utilizar bases de datos de imágenes de acceso libre, previamente usadas en trabajos del área o en competencias internacionales como la ImageCLEF. Además, nos basamos en métricas de calidad estándar para el área como Precisión, Recall, F-Measure, MAP, entre otras. Debido a que muchos trabajos no mencionan con detalle los parámetros o técnicas empleadas en sus experimentos, la tarea de comparación de resultados contra ellos se vuelve compleja. En este sentido, intentamos definir lo más detallado posible todo lo utilizado a la hora de la experimentación con el objetivo de facilitar la comparación contra este trabajo.

En cuanto a los histogramas de color, su extenso uso dentro del área de CBIR se debe a que el color es una de las características más intuitivas de las imágenes y que tienen buen desempeño en cuanto a complejidad espacial y temporal. El espacio de color más utilizado para los histogramas es el RGB, sin embargo el HSV también ha sido estudiado debido a que es perceptualmente más uniforme que el RGB. En nuestro trabajo, observamos que el espacio HSV consiguió mejorar los resultados de RGB entre un 15 % y un 40 % en todas las bases de datos utilizadas.

A diferencia de TBIR, en CBIR no se le ha dado importancia a funciones de *scoring* que permitan eliminar imágenes irrelevantes para la consulta en la lista de resultados devueltos. Con el propósito de limitar el número de imágenes devueltas, proponemos una función de *scoring* para los histogramas de color, con la cual se puede obtener listas de resultados de mayor calidad que balancean precisión y recall. Cabe destacar que este punto se estudia también desde su aspecto formal, demostrando que la función de *scoring* está correctamente definida para los histogramas de color.

Con respecto a las estructuras de índices en CBIR, se eligió estudiar la utilización de los SOM en este tipo de tareas. Partiendo del hecho que se requieren estructuras que permitan trabajar con grandes conjuntos de imágenes y descriptores de alta dimensionalidad, se decidió avanzar sobre los SOM y variantes del mismo que cumplieran con los anteriores requisitos. Se estudiaron modelos alternativos al SOM tradicional como BSOM, ParSOM, entre otros. En base al estudio de estas variantes, se propone un modelo alternativo que combina las características de ejecución en paralelo del ParSOM con el algoritmo *batch*: el *ParBSOM*. Este modelo reduce considerablemente los tiempos de entrenamiento y utilizando dos nodos de procesamiento se obtuvieron mejoras que alcanzan el 40 % con respecto al BSOM y que llegan a superar el 50 % con respecto al ParSOM. Vale aclarar que tanto ParSOM como BSOM ya mejoraban los tiempos del SOM tradicional.

En trabajos del área se han aplicado SOM y variantes a sistemas de CBIR, sin embargo poco se ha hecho para evaluar la calidad de recuperación de los SOM contra el algoritmo exacto, que consiste en una búsqueda lineal sobre la base de imágenes (enfoque al que llamamos Fuerza Bruta). Debido a que consideramos este aspecto de vital importancia para tener en cuenta a los SOM como candidatos para índices de CBIR, se decidió medir experimentalmente la calidad de los mapas generados así como también la diferencia de tiempos con respecto al algoritmo de Fuerza Bruta. Se entrenaron redes con las variantes del ParSOM y ParBSOM, presentando ambas un comportamiento similar en cuanto a calidad de resultados (a diferencia de los tiempos de entrenamiento, donde el ParBSOM resultó ser considerablemente más eficiente). Además, si comparamos estas variantes del SOM con respecto al algoritmo exacto, las pérdidas de calidad se mantienen menores al 10 % en todas las bases de datos utilizadas. En cuanto a tiempos, para la recuperación de una imagen en cada base de datos usando como índice a un SOM se obtienen mejoras superiores al 90 % con respecto al algoritmo de Fuerza Bruta.

Una vez estudiados estos aspectos relacionados al área de CBIR, decidimos poner foco en la recuperación híbrida. Se analizaron algunas de las variantes para la combinación de resultados provenientes de TBIR y CBIR que fueron presentadas recientemente en el área. Finalmente, se optó por utilizar la técnica de *refinamiento*, la cual consiste en ordenar los resultados de TBIR a partir de los de CBIR. La idea detrás de esta estrategia es quedarse con los resultados semánticamente relevantes para la consulta y ordenarlos por su similitud visual, dándole mayor importancia a los resultados de TBIR (la cual se encuentra actualmente más avanzada que CBIR). Se llevaron a cabo experimentos utilizando las técnicas estudiadas a lo largo de este trabajo (histogramas de color HSV, la función de scoring para eliminar imágenes irrelevantes, ParBSOM como índice y refinamiento para combinar los resultados), consiguiendo enriquecer los resultados devueltos por TBIR y obteniendo mejoras entre 10 % y 20 % para diferentes

métricas de calidad.

Como aporte adicional, se presenta un software al que denominamos *Envision* y que implementa todo lo estudiado a lo largo de este trabajo. Se caracteriza por su alta modularidad, lo que facilita la incorporación de nuevas técnicas para recuperar imágenes como pueden ser nuevos descriptores, índices, entre otras.

7.2. Trabajos futuros

Debido al gran crecimiento que viene experimentando el área de VIR durante los últimos años, existe una enorme cantidad de posibles caminos a seguir. Durante este trabajo, se ha dedicado la mayor parte del tiempo a avanzar sobre el área de CBIR, en particular sobre el tema de la indexación de los descriptores de las imágenes. Se eligió trabajar con histogramas de color dado que eran de los más populares en el área, sin embargo no son los únicos descriptores disponibles (tal como mencionamos en la Sección 2.3.1). Un posible trabajo a futuro es estudiar descriptores que se basen en la forma y/o textura y ver cómo es posible combinarlos con los histogramas de color para dar lugar a un descriptor más completo.

Otro aspecto que nos parece interesante es continuar la investigación relacionada a la recuperación híbrida. Por un lado, es escasa la cantidad de bases de datos disponibles que provean tanto información textual como visual y algunas bases todavía se encuentran en pleno desarrollo. Sobre este tema, un posible trabajo sería continuar con la búsqueda de bases de datos híbridas que permitan realizar más experimentos y analizar incluso la posibilidad de armar una propia. Por otro lado, consideramos que todavía es posible avanzar en las estrategias de combinación de resultados, siendo éste otro punto interesante para continuar a futuro.

Apéndice A

Demostraciones

A.1. Normalización de norma L1

En esta sección se demostrará que es posible normalizar la distancia que surge de comparar dos histogramas de color a través de la norma L1. Este resultado se utiliza en el Capítulo 3 para definir una función de scoring apropiada.

Sean H, H' histogramas de color (de n bins cada uno), queremos definir una función tal que:

$$\text{normalizar}(\mathbf{d}_{\mathbf{L1}}(H, H')) \in [0, 1]$$

Para lograr esto, vamos a encontrar una cota para la distancia entre dos histogramas y este resultado será usado para definir la función de normalización.

Por un lado, por construcción de los histogramas, sabemos que:

$$\sum_{i=1}^n H[i] = \sum_{i=1}^n H'[i] = 1 \quad (\text{A.1})$$

$$\forall i = 1 \dots n \quad H[i] \geq 0 \quad \text{y} \quad H'[i] \geq 0$$

Luego, por definición de la norma L1:

$$\mathbf{d}_{\mathbf{L1}}(H, H') = \sum_{i=1}^n |H[i] - H'[i]|$$

Ahora, enunciamos dos propiedades que usaremos luego. Suponiendo que a y b son ambos mayores a 0:

$$|a + b| = a + b \quad (\text{A.2})$$

$$|a - b| \leq |a + b| \quad (\text{A.3})$$

Entonces,

$$\begin{aligned}
\mathbf{d}_{\mathbf{L}1}(H, H') &= \sum_{i=1}^n |H[i] - H'[i]| && (\text{por def.}) \\
&\leq \sum_{i=1}^n |H[i] + H'[i]| && (\text{por A.3}) \\
&= \sum_{i=1}^n H[i] + \sum_{i=1}^n H'[i] && (\text{por A.2}) \\
&= 1 + 1 && (\text{por A.1}) \\
&= 2
\end{aligned}$$

Ahora, usando que $\mathbf{d}_{\mathbf{L}1}(H, H') \leq 2$, definimos la función de normalización como:

$$\text{normalizar}(\mathbf{d}_{\mathbf{L}1}(H, H')) = \frac{\mathbf{d}_{\mathbf{L}1}(H, H')}{2} \in [0, 1]$$

A.2. Complejidad temporal de SOM y BSOM

En esta sección se realizará una comparación de la complejidad temporal de los algoritmos de entrenamiento del SOM tradicional (versión *online*) y del BSOM (versión *batch*).

Dadas las siguientes variables:

$$\begin{aligned}
E &= \text{cantidad de épocas} \\
N &= \text{cantidad de patrones} \\
n &= \text{cantidad de neuronas} \\
m &= \text{dimensión de cada patrón}
\end{aligned}$$

Descripción: Entrenamiento *online*

entrenarOnline

Inicio

Para cada $t=1\dots E$

 Para cada $i=1\dots N$

$x = \text{patrones}[i]$

$\text{bm} = \text{buscarGanadora}(x)$

$\text{actualizarOnline}(\text{neuronas}[\text{bm}], x)$

 Fin para

Fin para

Fin

$$T_{\text{entrenarOnline}} = EN(\text{buscarGanadora} + \text{actualizarOnline})$$

Descripción: Búsqueda de la neurona ganadora

buscarGanadora(*Patrón x*)

Inicio

Para cada $j=1\dots n$

dist = distanciaEntreVectores(*x*,neuronas[j].vectorPesos)

Si dist < min

 bmu = j

 min = dist

Fin si

Fin para

Devolver bmu

Fin

La función de distancia entre vectores dependerá de la aplicación donde se use, pero asumimos que tiene m operaciones (la dimensión de los vectores).

$$T_{buscarGanadora} = nm$$

Descripción: Actualización *online*

actualizarOnline(*Neurona bmu, Patrón x*)

Inicio

Para cada $j=1\dots n$

 h = alfa * radio

 Para cada $k=1\dots m$

 neurona[j].vectorPesos[k] = neurona[j].vectorPesos[k] + h * (*x*[k] - neurona[j].vectorPesos[k])

 Fin para

Fin para

Fin

$$T_{actualizarOnline} = nm$$

Entonces,

$$\begin{aligned} T_{entrenarOnline} &= EN(\text{buscarGanadora} + \text{actualizarOnline}) \\ &= EN(nm + nm) \\ &= EN(2nm) \\ &= 2ENnm \end{aligned}$$

Luego, podemos decir que *entrenarOnline* tiene $\mathcal{O}(ENnm)$.

Descripción: Entrenamiento *batch*

entrenarBatch

Inicio

Para cada $t=1\dots E$

//Inicializo regiones de Voronoi

Para cada $j=1\dots n$

cantVoronoi[j]=0

Para cada $k=1\dots m$

voronoi[j][k] = 0

Fin para

Fin para

Para cada $i=1\dots N$

$x = \text{patrones}[i]$

bmú = buscarGanadora(x)

Para cada $k=1\dots m$

voronoi[bmú][k] = voronoi[bmú][k] + x[k]

Fin para

cantVoronoi[bmú]++

Fin para

actualizarBatch(voronoi,cantVoronoi)

Fin para

Fin

La descripción del algoritmo anterior se basa en la optimización que incluye *Regiones de Voronoi*, tal como figura en [35].

$$T_{\text{entrenarBatch}} = E(nm + N(\text{buscarGanadora} + m) + \text{actualizarBatch})$$

Descripción: Actualización *batch*

```

actualizarBatch(Double[ ][ ] voronoi, Double[ ] cantVoronoi)
Inicio
    Para cada j=1...n
        //Inicializo neurona
        Para cada k=1...m
            neurona[j].vectorPesos[k] = 0
        Fin para

        div=0
        Para cada j1=1...n
            h = radio
            Para cada k=1...m
                neuronas[j].vectorPesos[k] = neuronas[j].vectorPesos[k] + h * voronoi[j1][k]
                div = div + h * cantVoronoi[j1]
            Fin para
        Fin para

        Para cada k=1...m
            neuronas[j].vectorPesos[k] = neuronas[j].vectorPesos[k]/div
        Fin para
    Fin para
Fin

```

$$T_{actualizarBatch} = n(m + nm + m) = n(nm + 2m) = n^2m + 2nm$$

Entonces,

$$\begin{aligned}
T_{entrenarBatch} &= E(nm + N(buscarGanadora + m) + actualizarBatch) \\
&= E(nm + N(nm + m) + n^2m + 2nm) \\
&= E(nm + Nnm + Nm + n^2m + 2nm) \\
&= E(3nm + Nnm + Nm + n^2m) \\
&= ENnm + En^2m + ENm + 3Enm
\end{aligned}$$

Luego, podemos decir que *entrenarBatch* tiene $\mathcal{O}(ENnm)$.

Sin embargo, para ver la diferencia de constantes, analicemos la diferencia entre las funciones T de cada uno:

$$\begin{aligned}
T_{entrenarOnline} - T_{entrenarBatch} &= 2ENnm - (ENnm + En^2m + ENm + 3Enm) \\
&= ENnm - E(3nm + Nm + n^2m) \\
&= Em(Nn - (3n + N + n^2)) \\
&= Em(N(n - 1) - (n^2 + 3n)) \\
&= Em(N(n - 1) - ((n + 4)(n - 1) + 4)) \\
&= Em((n - 1)(N - (n + 4 + \frac{4}{n-1}))) \\
&= Em(n - 1)(N - n - 4 - \frac{4}{n-1})
\end{aligned}$$

Entonces, como $N - n - 4 - \frac{4}{n-1}$ será mayor a medida que la diferencia entre N y n se incremente y como $Em(n - 1) > 0$, podemos concluir que cuanta mayor sea la diferencia entre

N y n , mayor será la ventaja de la versión *batch*. Típicamente, sucede que $N \gg n$, por lo que el entrenamiento *batch* es considerablemente más eficiente.

Bibliografía

- [1] G. Arroyave, O. O. Lobo, and A. Marín, “A Parallel Implementation of the SOM Algorithm for Visualizing Textual Documents in a 2D Plane,” 2002.
- [2] J. Laaksonen, M. Koskela, S. Laakso, and E. Oja, “PicSOM—content-based image retrieval with self-organizing maps,” *Pattern Recogn. Lett.*, vol. 21, no. 13-14, pp. 1199–1207, 2000.
- [3] A. Rauber, D. Merkl, and M. Dittenbach, “The Growing Hierarchical Self-Organizing Map: Exploratory Analysis of High-Dimensional Data,” *IEEE Transactions on Neural Networks*, vol. 13, pp. 1331–1341, 2002.
- [4] M. Grubinger, *Analysis and Evaluation of Visual Information Systems Performance*. PhD thesis, School of Computer Science and Mathematics, Faculty of Health, Engineering and Science, Victoria University, Melbourne, Australia, 2007.
- [5] S. Haykin, *Neural Networks: A Comprehensive Foundation*. Prentice Hall, 2 ed., 1999.
- [6] C. Böhm, S. Berchtold, and D. A. Keim, “Searching in high-dimensional spaces: Index structures for improving the performance of multimedia databases,” *ACM Comput. Surv.*, vol. 33, pp. 322–373, September 2001.
- [7] T. Kohonen, “Self-organized formation of topologically correct feature maps,” *Biological Cybernetics*, vol. 43, pp. 59–69, 1982.
- [8] T. Kohonen, *Self-Organizing Maps*. Springer-Verlag, 3 ed., 2001.
- [9] P. Tomsich, A. Rauber, and D. Merkl, “parSOM: Using parallelism to overcome memory latency in self-organizing neural networks,” in *High Performance Computing and Networking*, pp. 61–5, Society Press, 2000.
- [10] P. Koikkalainen and E. Oja, “Self-organizing hierarchical feature maps,” *IJCNN International Joint Conference on Neural Networks*, vol. 2, pp. 279–284, 1990.
- [11] H. Shao, T. Svoboda, and L. van Gool, “ZuBuD — Zurich Buildings Database for Image Based Recognition,” tech. rep., Swiss Federal Institute of Technology, Switzerland, 2003.
- [12] G. Schaefer and M. Stich, “UCID - An Uncompressed Colour Image Database,” in *Storage and Retrieval Methods and Applications for Multimedia 2004*, vol. 5307 of *Proceedings of SPIE*, pp. 472–480, 2004.

- [13] D. Nistér and H. Stewénius, “Scalable Recognition with a Vocabulary Tree,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2, pp. 2161–2168, 2006.
- [14] M. Grubinger, P. Clough, A. Hanbury, and H. Müller, “Overview of the ImageCLEFphoto 2007 Photographic Retrieval Task,” *Advances in Multilingual and Multimodal Information Retrieval: 8th Workshop of the Cross-Language Evaluation Forum, CLEF 2007, Revised Selected Papers*, pp. 433–444, 2008.
- [15] T. Kato, “Database architecture for content-based image retrieval,” in *Proceedings of SPIE Image Storage and Retrieval Systems*, vol. 1662, (San Jose, CA, USA), pp. 112–123, 1992.
- [16] Y. Rui and T. S. Huang, “Image retrieval: Current techniques, promising directions and open issues,” *Journal of Visual Communication and Image Representation*, vol. 10, pp. 39–62, 1999.
- [17] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yanker, “Query by image and video content: the QBIC system,” *Computer*, vol. 28, no. 9, pp. 23–32, 1995.
- [18] P. P. Sclaroff, A. Pentl, R. W. Picard, and S. Sclaroff, “Photobook: Content-Based Manipulation of Image Databases,” *SPIE Storage and Retrieval Image and Video Databases II*, vol. 2185, 1994.
- [19] C. Carson, M. Thomas, S. Belongie, J. M. Hellerstein, and J. Malik, “Blobworld: A System for Region-Based Image Indexing and Retrieval,” in *Third International Conference on Visual Information Systems*, pp. 509–516, Springer, 1999.
- [20] S. Siggelkow, M. Schael, and H. Burkhardt, “SIMBA - Search IMages By Appearance,” in *Pattern Recognition, Proc. of 23rd DAGM Symposium, B. Radig and S. Floryczk, Eds. Sept. 2001, number 2191 in LNCS Pattern Recognition*, pp. 9–17, Springer Verlag, 2001.
- [21] T. Gass, T. Weyand, T. Deselaers, and H. Ney, “FIRE in ImageCLEF 2007: Support Vector Machines and Logistic Regression to Fuse Image Descriptors in for Photo Retrieval,” in *Advances in Multilingual and Multimodal Information Retrieval 8th Workshop of the Cross-Language Evaluation Forum, CLEF 2007*, vol. 5152 of *LNCS*, (Budapest, Hungary), Springer, 2008.
- [22] C. Frankel, M. J. Swain, and V. Athitsos, “WebSeer: An Image Search Engine for the World Wide Web,” tech. rep., University of Chicago, Chicago, IL, USA, 1996.
- [23] J. R. Smith and S.-F. Chang, “Visually Searching the Web for Content,” *IEEE MultiMedia*, vol. 4, no. 3, pp. 12–20, 1997.
- [24] M. J. Swain and D. H. Ballard, “Color indexing,” *International Journal of Computer Vision*, vol. 7, pp. 11–32, 1991.
- [25] C. D. Manning, P. Raghavan, and H. Schütze, *An Introduction to Information Retrieval*. Cambridge University Press, 2009.

- [26] J. Ruiz-del Solar and P. Navarrete, "FACERET: An Interactive Face Retrieval System Based on Self-Organizing Maps," in *CIVR '02: Proceedings of the International Conference on Image and Video Retrieval*, (London, UK), pp. 157–164, Springer-Verlag, 2002.
- [27] P. Prentis, "GalSOM - Colour-Based Image Browsing and Retrieval with Tree-Structured Self-Organising Maps," in *Proceedings of the 6th Int. Workshop on Self-Organizing Maps (WSOM 2007)*, (Bielefeld, Germany), 2007.
- [28] M. Bressan, "Categorization of Generic Images," in *Workshop de Imágenes*, (Buenos Aires, Argentina), 2008.
- [29] E. de Ves, A. Ruedin, D. Acevedo, X. Benavent, and L. Seijas, "A new wavelet-based texture descriptor for image retrieval," *12th International Conference on Computer Analysis of Images and Patterns, Lecture Notes in Computer Science*, vol. 4673/2007, pp. 895–902, 2007.
- [30] M. Koskela, *Interactive Image Retrieval using Self-Organizing Maps*. PhD thesis, Helsinki University of Technology, Espoo, Finland, 2003.
- [31] O. Jonsgård, "Improvements on colour histogram-based CBIR," Master's thesis, Gjøvik University College, Stockholm, Norway, 2005.
- [32] J. R. Smith and S.-F. Chang, "Single color extraction and image query," in *ICIP '95: Proceedings of the 1995 International Conference on Image Processing*, vol. 3, (Washington, DC, USA), p. 3528, IEEE Computer Society, 1995.
- [33] Z. L. Lin, L. Fuzong, and Z. Bo, "A CBIR Method Based On Color-Spatial Feature," in *Proc. IEEE Region 10 Annual International Conference 1999 (TENCON'99)*, (Cheju, Korea), pp. 166–169, 1999.
- [34] B. S. Manjunath, J. R. Ohm, V. V. Vasudevan, and A. Yamada, "Color and Texture Descriptors," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 11, no. 6, pp. 703–715, 2001.
- [35] J. Vesanto, J. Himberg, E. Alhoniemi, and J. Parhankangas, "SOM Toolbox for Matlab 5," tech. rep., Helsinki University of Technology, Finland, 2000.
- [36] J. Vesanto, "SOM algorithm implementation in SOM Toolbox." <http://www.cis.hut.fi/projects/somtoolbox/documentation/somalg.shtml>, 2005.
- [37] B. Fritzke, "Growing Grid - a self-organizing network with constant neighborhood range and adaptation strength," *Neural Processing Letters*, vol. 2, no. 5, pp. 9–13, 1995.
- [38] I. Ounis, G. Amati, V. Plachouras, B. He, C. Macdonald, and C. Lioma, "Terrier: A High Performance and Scalable Information Retrieval Platform," in *Proceedings of ACM SIGIR'06 Workshop on Open Source Information Retrieval (OSIR 2006)*, 2006.
- [39] C. Lioma, C. Macdonald, V. Plachouras, J. Peng, B. He, and O. I., "University of Glasgow at TREC 2006: Experiments in Terabyte and Enterprise Tracks with Terrier," in *Proceedings of the 15th Text REtrieval Conference (TREC 2006)*, 2006.