Huya: un Sistema para Recuperación de Imágenes basado en MRML

Robinson S. Rivas-Suárez, Yeny Hernández
Universidad Central de Venezuela, Escuela de Computación
Centro de Computación Paralela y Distribuida
Caracas 1040, Venezuela
{rrivas,yhernand}@strix.ciens.ucv.ve

Resumen

En la actualidad existen numerosos sistemas que permiten recuperar imágenes en base al contenido de las mismas, llamados genéricamente Content-Based Image Retrieval (CBIR). Algunos de estos sistemas se basan en la arquitectura Cliente/Servidor, sin embargo no existen estándares aceptados por la comunidad de investigación para comunicar diferentes sistemas CBIR Cliente/Servidor. Para resolver este problema, se introdujo el protocolo MRML, basado en XML y diseñado para permitir la comunicación entre diferentes sistemas CBIR independientemente de los algoritmos, estructuras de datos y tecnologías utilizadas para su implementación.

En este trabajo, se diseñó e implementó un servidor MRML utilizando tecnologías portables, de acuerdo a la especificación 1.0 del lenguaje. Sin embargo esta especificación no resuelve todas las complejidades de la interacción de los sistemas CBIR, por lo que en este trabajo se definieron algunas extensiones a dicha especificación. Finalmente se presenta el sistema desarrollado así como pruebas de su funcionalidad, y propuestas para nuevas extensiones del protocolo.

Palabras clave: MRML, Recuperación de Imágenes Basada en Contenido, XML, Sistemas Basados en WEB

Abstract

Actually there are many systems that search on Images Databases based on the image's content or meaning, usually referred to as Content Based Image Retrieval systems (CBIR systems). Some of these CBIR systems are designed using a Client/Server architecture, so it is in theory possible to communicate different kinds of clients and servers once they share the minimum protocols. However, there is not a well accepted standard protocol to satisfy that objective. To achieve this goal it was proposed MRML as an open XML specification useful for any CBIR system based on Client/Server architecture.

Since the first specification (MRML version 1.0) did not fulfill all the complexities of web-based CBIR systems, we proposed some new ideas and extensions for the protocol, such as indexes vectors specifications, both local and remote images search and sessions managing. Some of these new ideas were implemented and tested in a Java MRML server, the first as far as we know. In this work we present the extensions proposed for MRML and a brief discussion about the differences of our proposals and those of MRML specification team.

Key Words: MRML, Content-Based Image Retrieval, XML, Web Based Systems

1. Introducción

Los sistemas de búsqueda de imágenes basada en contenido (CBIR - Content Based Image Retrieval) se han popularizado debido principalmente al crecimiento de la Internet, lo que ha puesto a disposición de la comunidad académica y empresarial millones de imágenes en los más diversos ámbitos de aplicación. La mayoría de los sistemas que se han creado utilizan algoritmos, técnicas de búsqueda, estructuras de datos y modelos de Bases de Datos propietarios, lo que ha dificultado la interoperabilidad entre los mismos, y por tanto la adecuada validación de las técnicas utilizadas y la interacción entre grupos de investigación y desarrollo **[1, 4]**.

En vista de esta situación, diversos autores han presentado propuestas para definir un protocolo que permita la interacción entre los sistemas CBIR disponibles en la Internet. Uno de estos protocolos es el MRML (Multimedia Retrieval Markup Language) definido y presentado por Henning Müeller y Stéphane Marchand-Maillet **[8]**. Este protocolo se basa en XML y permite definir la mayoría de las interacciones necesarias para establecer comunicación, definir requerimientos y obtener respuestas desde servidores de objetos multimedia, particularmente imágenes. La idea principal es que se puedan independizar los diferentes elementos de los que consta un sistema CBIR Cliente/Servidor en una intranet o basado en Web.

Sin embargo, no son muchos los desarrollos que se han hecho basados en este protocolo. Aparte de las especificaciones y pruebas de los autores **[8,9]** y del proyecto GIFT de la Free Software Foundation **[3]** no existen clientes y servidores comerciales basados en MRML. En este trabajo, se presenta el sistema HUYA desarrollado en la Universidad Central de Venezuela, que consta de un servidor MRML y un cliente de prueba, que permite implementar las principales funcionalidades del protocolo en su versión 1.0. Durante el desarrollo del sistema, se vio la necesidad de aumentar las funcionalidades del protocolo, por lo que se han hecho propuestas en esa dirección. En particular, la especificación implementada añade operaciones para la transmisión de *vectores característicos* de imágenes y la búsqueda en repositorios de imágenes tanto locales como remotos.

El sistema desarrollado se implementó usando Java y herramientas de software abierto, y puede ser ejecutada en cualquier máquina donde haya una JVM. Se hicieron pruebas de funcionamiento sobre colecciones de imágenes indexadas manualmente y automáticamente, y sobre colecciones de imágenes locales y remotas al cliente, usando varias estructuras de datos multidimensionales para la indexación de las imágenes.

El trabajo consta de cinco secciones, incluyendo esta introducción. En la sección 2 se discuten las características principales de los sistemas CBIR, en cuanto sistemas de almacenamiento, indexación y búsqueda de información multimedia. En la sección 3 se muestra una panorámica del lenguaje MRML, sus principales características y las especificaciones de la versión 1.0. Igualmente se presentan algunas de las modificaciones que se hicieron a la especificación.

La sección 4 muestra la arquitectura del sistema desarrollado y el ambiente de desarrollo y prueba. Por último, la sección 5 presenta los resultados y conclusiones obtenidos hasta el momento en este proyecto, así como una perspectiva de los trabajos en desarrollo y futuros.

2. Sistemas CBIR

Por sistemas CBIR se entiende una serie de tecnologías que permiten al usuario almacenar, indexar, clasificar y posteriormente recuperar información de imágenes digitales **[4]**. Debido a que los sistemas CBIR se encargan de varias actividades anteriores y posteriores a la recuperación de las imágenes, es común caracterizarlos como sistemas modulares donde cada uno de los módulos se encarga de una o varias primitivas de procesamiento, entre las que destacan:

1.- **Extraer características de las imágenes:** Lo que se hace de forma automática o semi-automática, de manera que las características sirvan para generar índices que faciliten la posterior clasificación y búsqueda.
2.- **Almacenar los índices:** para lo cual se utilizan Estructuras de Datos y manejadores de Bases de Datos especializados para el manejo de información multidimensional, que es como usualmente se define la información proveniente de imágenes y otros objetos Multimedia.
3.- **Construir las solicitudes de búsqueda:** de acuerdo a un patrón o patrones proporcionados por el usuario o sistema cliente. Por lo general al tratarse de imágenes, se utiliza el paradigma llamado

Búsqueda por Muestra o QBE (Query by Example) en la que el usuario o sistema cliente presenta una imagen que sirve de modelo para la búsqueda, y el sistema retorna las imágenes más parecidas de acuerdo a las métricas asociadas a las estructuras de datos y algoritmos especializados.

4.- **Retornar los resultados del proceso de búsqueda:** de acuerdo a especificaciones definidas por el motor de búsqueda o por el cliente.

Puede observarse que los diferentes pasos en la construcción y funcionamiento de los sistemas CBIR no indican nada acerca de la arquitectura del sistema. Se puede conceptualizar dichos sistemas como sistemas centralizados donde los diferentes pasos de funcionamiento los ejecuta un solo sistema modular, o como sistemas distribuidos en los que cada operación la realiza un elemento distribuido.

Para cada uno de los pasos descritos anteriormente se han desarrollado numerosas técnicas, algoritmos y arquitecturas como se describirá brevemente de inmediato.

1.1. Modelos de Extracción de Características

Las características que se pueden extraer de una imagen se pueden clasificar en varias categorías de acuerdo al nivel de abstracción y a las semánticas asociadas a las características mismas. Según Eakins **[1]** normalmente se definen tres niveles de extracción de características de imágenes:

- **Características Primitivas**: son aquellas basadas exclusivamente en la información numérica asociada a la imagen digital, por ejemplo brillo, luminosidad, etc.
- **Características Lógicas**: son aquellas que representan elementos geométricos, morfológicos o pictóricos que requieren de la aplicación de técnicas especializadas sobre la información numérica, por ejemplo, la detección de rostros, la presencia de ciertas formas o figuras dentro de una imagen, etc.
- **Características Abstractas**: Son aquellas que representan valores abstractos de la realidad. En general, dependen de técnicas muy avanzadas relacionadas con Inteligencia Artificial, por ejemplo la detección de emociones humanas, categorías políticas, deportivas, etc.

La extracción de características se puede hacer de forma manual, semi-automática o automática. En **[15, 6]** se presentan ejemplos de sistemas que emplean diferentes tipos de técnicas para la extracción de características.

1.2.- Estructuras de Datos Multidimensionales

Una vez extraídas las características de las imágenes, éstas deben ser codificadas de acuerdo a algun criterio y almacenadas en estructuras de datos que permitan su rápida búsqueda y manipulación. En general, de cada imagen se obtienen múltiples valores que representan en su conjunto una coordenada de un espacio N-dimensional. Muchas estructuras de datos se han definido para almacenar esta información multidimensional, tales como R-Trees, Q-Trees, SR-Trees, KD-Trees, etc.

Las diferencias entre las estructuras de datos vienen dadas por su capacidad para almacenar distintos tipos de información, almacenar información redundante o incompleta, hacer búsquedas por regiones, etc. Las estructuras de datos están asociadas además a los algoritmos requeridos para realizar las consultas, en especial los llamados algoritmos de búsqueda de los vecinos más próximos *(nearest neighbor searching)*. Estos algoritmos permiten que, dada una imagen de referencia, se extraigan de la estructura de datos aquellas imágenes que más se asemejen a la referenciada de acuerdo a algunas métricas.

1.3. Construcción de las Solicitudes

Se han definido algunos lenguajes para indicar cómo un cliente debe especificar lo que desea buscar en un ambiente de recuperación de información multimedia, específicamente imágenes. Aun cuando a la fecha no se tiene un estándar universalmente aceptado, sí se han presentado diferentes propuestas para construir dichas solicitudes **[2]**. Algunas propuestas manejan lenguajes de consulta textuales como MMDOC-QL **[7]**. Otras se basan en la construcción de prototipos de imágenes como base de la búsqueda como QBIC de IBM **[11]**, mientras que otros se basan en la presentación de una o más imágenes de referencia, usando la técnica conocida como Búsqueda por Muestra (*Query by Example*) **[6,10,15]**.

La forma como se especifican las características del objeto a consultar es independiente de la forma como se realiza

dicha consulta, ya que las técnicas de búsqueda dependen de los algoritmos asociados a las Estructuras de Datos de almacenamiento. En general, en ambientes Cliente/Servidor, la especificación de la búsqueda se encuentra del lado del cliente, mientras que los algoritmos de búsqueda y las estructuras de almacenamiento se encuentran del lado del servidor.

1.4.- Retorno de los resultados

La última fase se limita a la presentación de las búsquedas efectuadas por el usuario. La forma, número de resultados, niveles de relevancia, información adicional que se presente a los clientes y otras consideraciones dependen de las especificaciones de las interfaces de usuario, y están en correspondencia directa con las capacidades de los algoritmos de búsqueda, las estructuras de almacenamiento y los lenguajes de especificación.

Los pasos explicados anteriormente se efectúan independientemente de la arquitectura de software, ya sea centralizada, distribuida o Cliente/Servidor. Sin embargo, en el caso de sistemas Cliente/Servidor (ver figura 2) se hace necesario independizar los el funcionamiento de los módulos Clientes y Servidores mediante el uso de un protocolo de comunicaciones que permita la interacción de diferentes clientes con diferentes servidores sin importar el lenguaje de programación, el Sistema Operativo, las plataformas de hardware, etc, tal como se tiene para plataformas maduras basadas en web (protocolos http, ftp o telnet). En la sección siguiente se muestra una propuesta funcional para lograr este objetivo.

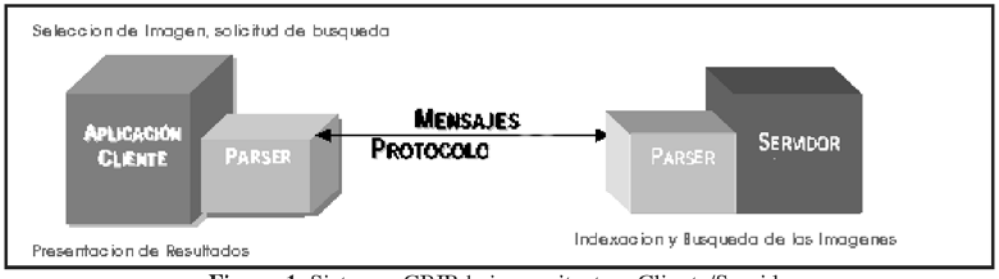

Figura 1: Sistemas CBIR bajo arquitectura Cliente/Servidor

3. El lenguaje MRML

El lenguaje de marcado para recuperación multimedia MRML (*Multimedia Retrieval Markup Language*) fue diseñado por Wolfgang Müller, Henning Müller y Stéphane Marchand-Meillet de la Universidad de Ginebra **[8]** como una forma de independizar los clientes y servidores en sistemas CBIR. MRML está basado en XML y define las principales interacciones que se requieren para establecer comunicaciones entre dos módulos del sistema CBIR, entre ellas establecer conexión, iniciar sesiones, obtener información sobre capacidades del servidor, enviar requerimientos al servidor y retornar respuestas al cliente. Un diagrama de estados de MRML se muestra en la figura 2.

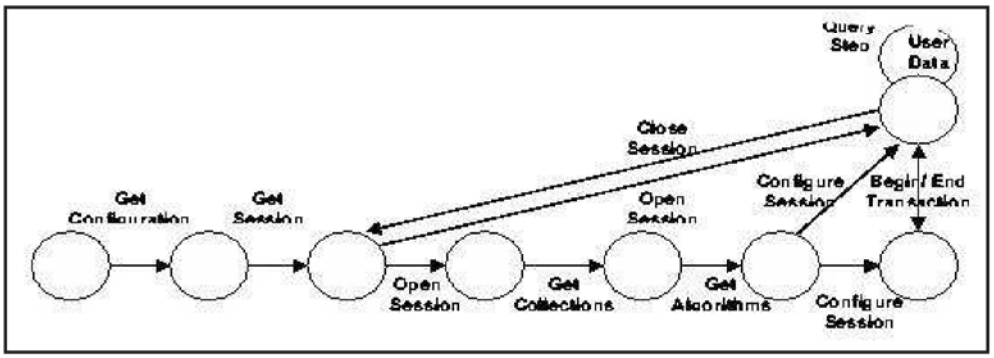

Figura 2: Diagrama de Transición de Estados de MRML

La comunicación Cliente-Servidor en MRML es una secuencia de conexiones. En cada conexión una petición o un conjunto de peticiones son respondidas usando un mensaje simple o un pequeño grupo de mensajes. La máquina de estados describe la comunicación, comenzando en el punto donde se hace el primer contacto con el servidor.

El cliente establece el primer contacto con el servidor enviando el mensaje de solicitud *get-server-properties*. Como respuesta el cliente recibe un mensaje *server-properties*, que en la especificación MRML 1.0 se define como vacío. Sin embargo este mensaje es importante si se piensa en la extensibilidad que ofrece el MRML para desarrollos futuros. Después de recibir la información de la descripción del servidor el cliente podrá enviar una petición para obtener una lista de las sesiones del usuario, usando la etiqueta *get-sessions*.

El mensaje de respuesta a esto es el *session-list*. El cliente podrá ahora abrir una sesión a través del mensaje *open-session*, obteniendo un *acknowledge-session-op* como respuesta. La sesión abierta es requerida para tener control sobre el manejo de estados, para poder realizar posteriores consultas. Abriendo la sesión, el servidor habrá recibido la identificación de usuario, la contraseña y la identificación del usuario, de existir en el sistema.

Todo lo anteriormente expuesto se necesita para saber qué colecciones y qué algoritmos el usuario puede ver. Después de obtener un identificador de sesión y una sesión abierta el cliente tiene la posibilidad de consultar tanto las colecciones como los algoritmos disponibles para cada colección. Las formas de consulta sobre los elementos anteriormente nombrados son dependientes del paradigma de consulta establecido. En particular un algoritmo puede contener un atributo como un identificador que lo una al tipo de colección en que puede ser usado (por ejemplo, algoritmos de similaridad de rostros solo podrían ser usados sobre colecciones de imágenes de rostros). Teniendo acceso a la lista de algoritmos y colecciones, el cliente tiene acceso a suficiente información para poder configurar la sesión que ya ha sido establecida.

Después de esto la sesión está totalmente configurada y pueden efectuarse las consultas (mediante el mensaje *query-step*). Las consultas pueden ser agrupadas dentro de un grupo de transacciones. El cliente también debe ser capaz de poder enviar los datos de usuario *user-data* al servidor. Estas etiquetas con datos del usuario contienen información sobre el usuario y se especifican para propósitos de aprendizaje por parte del servidor.

Al solicitar una consulta al servidor, la respuesta es un mensaje *query-result*. La especificación original MRML indica que los objetos de la colección se especifican mediante un URL, de manera que la localización real de los objetos puede ser arbitraria. Sin embargo, esta simplificación también significa que los clientes no pueden enciar las características de los objetos que desean consultar al servidor sino únicamente los URL, lo que hace que todo el trabajo de indexación y extracción de características recaiga sobre el servidor una vez dispone de los URL de consulta.

Los mensajes MRML 1.0 se especifican mediante un DTD XML donde se detallan los tipos de mensajes y la construcción semántica de MRML. En la figura 3 se muestra un extracto del DTD de MRML en el que se muestran los tipos de mensaje existentes. Obsérvese que no hay especificaciones de seguridad, búsquedas locales o remotas, mantenimiento de colecciones, etc.

```
<!ELEMENT mrml (begin-transaction?,
  (get-configuration
  |configuration-description
  |get-sessions
  |session-list
  |open-session
  |rename-session
  |close-session
  |delete-session
  |get-collections
  |collection-list
  |get-algorithms
  |algorithm-list
  |configure-session
  |query-step
  |query-result
  |user-data
  |error)?,
  end-transaction?)>
<!ATTLIST mrml
   session-id CDATA #IMPLIED
   transaction-id CDATA #IMPLIED
   mrml-id ID #IMPLIED>
```

Figura 3: Extracto del DTD de MRML 1.0, con los tipos demensaje soportados

El DTD, la especificación formal de MRML 1.0 y documentación al respecto se encuentran en el sitio oficial de

MRML (http://www.mrml.net). Dos ejemplos de mensaje MRML se muestran en las figuras 4 y 5.

```
<!-- OPEN-SESSION: ESTABLECE COMUNICACIÓN CON EL SERVIDOR
-->
<?XML VERSION="1.0" ENCODING="UTF-8"?>
<!DOCTYPE MRML SYSTEM "HTTP://150.185.75.106/HUYA/MRML.DTD">
<MRML SESSION-ID="2003-10-09 20:23:30.494">
<OPEN-SESSION USER-NAME="YENYCAROL" SESSION-NAME="OPEN-SESSION" />
</MRML>
```

Figura 4: Mensaje de establecimiento de sesión.

```
<?XML VERSION="1.0" ENCODING="UTF-8"?>
<!DOCTYPE MRML SYSTEM "HTTP://150.185.75.106/HUYA/MRML.DTD">
<MRML SESSION-ID="2003-10-09 20:23:30.494" >
<GET-COLLECTIONS>
<COLLECTION
COLLECTION-LOCATION-STRING="HTTP://150.185.75.106/HUYA/AMUSEMENT/"/>
</GET-COLLECTIONS>
</MRML>
```

Figura 5: Mensaje de solicitud de colecciones

4. El Sistema HUYA

En base a la versión 1.0 de MRML, se inició en la Escuela de Computación de la Universidad Central de Venezuela la construcción de un sistema Cliente/Servidor basado en Java que sirviera de plataforma de prueba para el protocolo. El sistema independiza las operaciones y definiciones del Cliente y el Servidor, que solamente interactúan por medio del pase de mensajes MRML y operan contra colecciones de datos que se acceden a través de un servidor Web, que puede estar independiente tanto del cliente como del servidor. El nombre del sistema se colocó como momenaje a un planeta extra-solar descubierto por un astrofísico Venezolano durante el año 2003. La figura 6 muestra la arquitectura del sistema Huya.

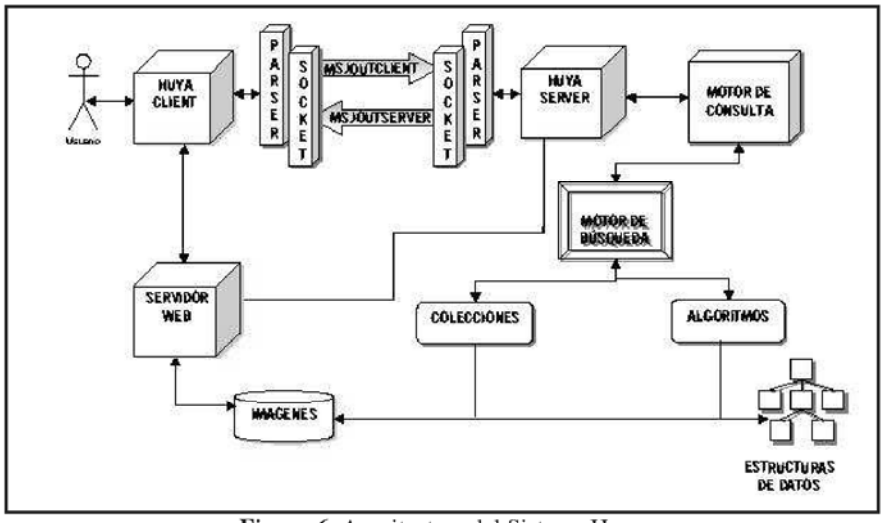

Figura 6: Arquitectura del Sistema Huya

4.1. Esquema de Interacción Cliente/Servidor

El sistema Huya se basa en el modelo solicitud-respuesta, en el que cada mensaje es tratado como una conexión individual, de manera que no hay una conexión abierta permanente entre el Cliente y el Servidor. Los pasos de la interacción del sistema se detallan a continuación:

1.- **Obtener Configuración:** Inicialmente el cliente establece una conexión con el servidor MRML en escucha, al establecer la conexión, el Cliente requiere conocer las propiedades de búsqueda y las colecciones de imágenes que posee el Servidor.

2.- **Enviar la Descripción de la Configuración:** El Servidor responde con las propiedades establecidas en su archivo de configuración, de manera tal que envía un mensaje al Cliente, con los Algoritmos de Búsqueda que posee y las Colecciones de Imágenes de respuesta. Las imágenes se encuentran físicamente en repositorios accesibles mediante el Servidor Web.

3.- **Obtener Imágenes de una Colección:** Si el Cliente desea conocer las imágenes de muestra de una colección dada, envía al servidor un mensaje, indicando el nombre de la colección seleccionada

3.1.- **Obtener los URL's de las Imágenes de la Colección:** Dado el nombre de una colección, el servidor consulta al servidor web y construye una lista con los URL's de las imágenes contenidas en la colección.

3.2.- **Retornar la Lista con las Imágenes de la Colección:** Esta actividad muestra la interacción realizada entre el servidor HuyaServer y el servidor web. En ese momento el servidor posee una lista de URL's de todas las imágenes de muestra.

3.3.-**Enviar la Lista al Cliente:** La lista de los URL's de las imágenes de muestra, es enviada al Cliente.

3.4.-**Solicitar las Imágenes al Servidor Web:** Una vez obtenida la lista de los URL's de las imágenes de muestra, el Cliente realiza una petición al Servidor web.

3.5.- **Retornar Imágenes**: El Servidor web retorna las imágenes consultadas por el Cliente.

4.- **Iniciar las Consultas:** Una vez obtenidas las imágenes de muestra, el cliente está en capacidad de realizar una o más consultas al servidor HuyaServer, en cualquiera de las dos formas predefinidas (ver sección 4.2).

4.1.- **Si el cliente desea realizar una consulta sobre una imagen local**, éste debe poseer el vector de características de la imagen seleccionada, de manera tal que pueda enviar este vector al servidor para que ejecute la consulta. Junto con el vector característico de la imagen seleccionada, se envía el nombre del algoritmo de búsqueda y la cantidad de imágenes que requiere como resultado.

4.1.1.- **Consultar las Estructuras de Datos Multidimensionales:** Dado el vector de características de la imagen, se hace un llamado al Algoritmo de Búsqueda seleccionado por el cliente que resuelve la búsqueda generando tantas imágenes resultado como el cliente haya solicitado.

4.2.- **Enviar el URL de la Imagen en caso que la búsqueda sea Remota:** Si el cliente desea realizar una consulta sobre una imagen de una de las colecciones del servidor, simplemente selecciona el URL de la imagen y se envía un mensaje de consulta junto con el nombre de un algoritmo de búsqueda y la cantidad de imágenes que requiere como resultado.

4.2.1.- **Buscar el Vector Característico y Consultar las Estructuras de Datos:** Si el cliente seleccionó una búsqueda remota (imagen de una colección del servidor), el servidor debe ser capaz de buscar el Vector Característico de la imagen de muestra y realizar el correspondiente llamado al Algoritmo de Búsqueda seleccionado por el Cliente.

5.- **Obtener Resultados de las Estructuras de Datos:** La estructura de datos y el Algoritmo seleccionados devuelven una lista con todas las imágenes resultado, esta lista es utilizada por el Servidor para enviar respuesta al Cliente.

6.- **Enviar la Lista de URL de Imágenes:** Una vez obtenidas las imágenes de respuesta, se envía una lista de los URL's de dichas imágenes al Cliente.

7.- **Consultar al Servidor Web:** Una vez el Cliente obtiene la lista de los URL's de las imágenes resultado, se realiza una petición al Servidor web para su presentación al usuario.

8.- **Retornar Imágenes y Mostrar Resultados:** El Servidor web retorna las imágenes consultadas por el Cliente, y éste hace la presentación al usuario.

4.2. Extensiones a MRML 1.0

Durante el desarrollo del proyecto e observó que las interacciones de los sistemas CBIR son más complejas que los mensajes disponibles en la especificación 1.0, por lo que se decidió añadir algunas observaciones al protocolo original. Estas observaciones han sido discutidas con los autores de MRML, que están actualmente definiendo el *draft* de la versión 2.0.

En particular, se vió la necesidad de implementar mensajes para diferenciar las búsquedas *locales* de las búsquedas *remotas*. La diferencia básica entre estos dos tipos de búsqueda es que las búsquedas *locales* permiten al cliente el envío del *vector característico* de una imagen de alguna colección local al cliente, en lugar de enviar el URL de la imagen. La ventaja de esto radica en que se pueden utilizar algoritmos de extracción de características diferentes a los definidos en el lado del servidor, e incluso pueden ser vectores característicos de imágenes sintéticas generadas por el usuario. El query remoto implica comparar imágenes localizadas en las colecciones del lado del servidor, identificándolas mediante su URL. En cualquiera de los dos casos, la comparación se hace sobre los algoritmos y colecciones ofertados por el servidor.

Los nuevos mensajes definidos se ejemplifican mediante las figuras 7 y 8. En ellas se muestra la especificación de los nuevos mensajes *query-local* y *query-remote.*

```
<!-- QUERY_LOCAL: ENVIAR LOS PARAMETROS DE CONSULTA SOBRE UNA
IMAGEN LOCAL SELECCIONADA
-->
<?XML VERSION="1.0" ENCODING="UTF-8"?>
<!DOCTYPE MRML SYSTEM "HTTP://150.185.75.106/HUYA/MRML.DTD">
<MRML SESSION-ID="2003-10-09 20:23:30.494" >
<QUERY-LOCAL>
<IMAGE NAME-IMAGE="AMU001.JPG" IMAGE-VECTOR="20;23;30;49;4"/>
<CANT-IMAGE NUMBER="4" />
<ALGORITHM ALGORITHM-NAME-STRING="RTREE" />
</QUERY-LOCAL>
</MRML>
```

Figura 7: Mensaje *query-local*

```
<!-- QUERY_REMOTE: ENVIAR LOS PARAMETROS DE CONSULTA SOBRE UNA
IMAGEN DE UNA COLECCIÓN SELECCIONADA
-->
<?XML VERSION="1.0" ENCODING="UTF-8"?>
<!DOCTYPE MRML SYSTEM "HTTP://150.185.75.106/HUYA/MRML.DTD">
<MRML SESSION-ID="2003-10-09 20:23:30.494" >
<QUERY-REMOTE>
<IMAGE IMAGE-LOCATION-
STRING="HTTP://150.185.75.106/HUYA/BOATS/BOAT01.JPG"/>
<CANT-IMAGE NUMBER="6" />
<ALGORITHM ALGORITHM-NAME-STRING="RTREE" />
<COLLECTION COLLECTION-NAME-STRING="BOATS" />
</QUERY-REMOTE>
</MRML>
```

Figura 8: Mensaje *query-remote*

Obsérvese que en el mensaje *query-local* se envía un vector característico llamado *image-vector* que se obtiene mediante la aplicación de algún algoritmo de extracción de características del lado del cliente. En el caso del mensaje *query-remote*, solamente se envía el URL de la imagen. En ambos casos, la colección donde se buscará la imagen, y el algoritmo de búsqueda se toman de las listas proporcionadas previamente por el Servidor.

4.3. Interfaces de Usuario y Detalles de Funcionamiento del Sistema Huya

A continuación se muestran las Interfaces de Usuario del Sistema Huya. En la figura 9, se muestra la consola del Servidor. Aunque este tipo de consola no es estrictamente parte del Servidor, es importante para efectos de depuración del sistema, y para tener un mejor control de la herramienta.

Figura 9: Consola de Administración de HuyaServer

La intefaz principal del Cliente se muestra en la figura 10. Obsérvese que antes de iniciar la interacción con el Servidor Huya, el Cliente tiene acceso a una o más colecciones de imágenes locales, que obtiene desde un archivo

de configuración XML. Para la implementación del sistema se utilizó el puerto IP 1251, aunque esto no es parte del estándar del protocolo y puede utilizarse cualquiera disponible.

Figura 10: Interfaz de Usuario del Cliente Huya y Ventana de Conexión

Las imágenes de las colecciones de prueba varían en cuanto a temática y formato, incluyendo imágenes de animales, paisajes, botes, flores y escenas de moda. Los vectores característicos se extrajeron de forma semiautomática, e incluyen características pictóricas y de forma (color, brillo, cantidad de objetos, disposición de los objetos, etc). Una descripción de estas características se encuentra en **[13,14]**

Una vez selecionada una forma de interacción, el usuario puede buscar imágenes de las dos formas predeterminadas (Búsqueda Local y Búsqueda Remota). La similitud de las imágenes depende del Algoritmo de Búsqueda Seleccionado, en los experimentos realizados se utilizó la Distancia Euclidiana sobre los vectores multidimensionales, almacenando los vectores característicos en un KDTree. En las figuras 11 y 12 se muestra un ejemplo de resultado obtenido en cada uno de estos tipos de búsqueda.

Figura 11: Resultado de una Búsqueda Local.

Figura 12: Resultado de una Búsqueda Remota

Es importante destacar que el sistema se programó utilizando Interfaces de Java, para lo cual se definió un API conciso que incluye**:**

- **IndexTreeInterface:** Interfaz para las Estructuras Multidimensionales soportadas por el Servidor.
- **FeatureExtractorInterface:** Interfaz que define el comportamiento de cualquier Algoritmo de Extracción de Caracterísitcas soportado tanto por el Cliente como por el Servidor.
- **FeatureInterface:** Interfaz que define cómo se estructuran las características de cada imagen.
- **CollectionInterface:** Interfaz que permite definir el comportamiento de las Colecciones de Imágenes, y permite encapsular la creación y manipulación de colecciones enteras de imágenes
- **RepositoryInterface:** Permite modelar el comportamiento de repositorios de imágenes. Cada repositorio consiste de: 1) una Estructura Multidimensional para el almacenamiento de los Vectores Característicos de las imágenes, 2) un Algoritmo de Extracción de dichos Vectores Característicos y 3) una Colección de Imágenes.

De esta forma se logró un sistema modular, en el que se pueden añadir nuevos Algoritmos de Extracción de Características y Estructuras de Datos Multidimendionales si se desea, siempre que se programe en Java respetando la Interfaz correspondiente. Para que estas nuevas implementaciones tengan efecto en el Cliente o el Servidor, basta añadirlos dentro del archivo de configuración XML correspondiente. En la figura 13 se muestra un ejemplo de archivo de configuración correspondiente al Servidor.

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
  <name>Huya-Server</name>
  <description>Servidor MRML de la UCV</description>
  <port>1251</port>
  <image-directory>http://ip:port/</image-directory>
  <dtd-url>http://ip:port/dtd/mrml1.0.dtd</dtd-url>
  <collections>
     <collection>
        <collection-name>animals</collection-name>
        <collection-id>0</collection-id>
        <collection-location>http://ip:port/name</collection-location>
     </collection>
  </collections>
  <algorithms>
     <algorithm>
        <algorithm-name>kdtree</algorithm-name>
        <algorithm-id>id_kdtree</algorithm-id>
        <algorithm-type>type_kdtree</algorithm-type>
        <algorithm-location>mrml.kdtree.KDTree</algorithm-location>
     </algorithm>
  <extractors>
     <extractor>
        <extractor-name>Simple Feature Extractor</extractor-name>
        <extractor-location>mrml.api.SimpleFeatureExtractor</extractor-location>
     </extractor>
  </extractors>
</configuration>
```

Figura 13: Ejemplo de Archivo de Configuración del Servidor

Una descripción completa del trabajo preliminar de Huya se encuentra en **[5,14]**.

5. Resultados y Trabajos Futuros

El trabajo realizado permitió obtener los siguientes resultados:

1. Se implementó la especificación 1.0 de MRML de forma modular, de manera que permite interactuar un Cliente contra un Servidor de forma transparente en cuanto a los detalles internos de cada uno de los componentes.
2. Como limitación del trabajo, no se implementaron los mensajes correspondietnes a *user-relevance*, esto es, mensajes diseñados específicamente para permitir que el Sevidor pueda aprender de la interacción del Cliente. Este tipo de aprendizaje está presente en sistemas como PicSOM **[10]** en el cual de las sucesivas búsquedas del Cliente se puede inferir el tipo de respuesta esperada.
3. La especificación de MRML permite la interacción sin la definición explítcita de sesiones, por lo cual éstas no fueron implementadas.
4. Se definieron tres nuevos nuevos mensajes:
   4.1 *collection-list*: permite obtener la lista de todas las imágenes de una colección específica.
   4.2 *query-remote:* permite solicitar resultados en base a una imagen remota, esto es, definida en las colecciones del servidor. Equivale al mensaje *query-step* de la especificación 1.0
   4.3 *query-local*: permite solicitar resultados en base a un Vector Característico provisto por el Cliente, y obtenido de algún algoritmo de extracción de características.

   Los nuevos mensajes se implementaron respetando la estructura sintáctica de MRML, y para su implementación se hizo necesario redefinir parcialmente el DTD de MRML 1.0.
5. La implementación mediante el uso de Java Interfaces ha permitido tener un software modular y fácil de actualizar, ya que los principales componentes del sistema pueden ser añadidos mediante su simple inclusión en archivos de configuración.

Durante el desarrollo del trabajo la interacción con el grupo original que definió MRML permitió encontrar algunas inconsistencias de la especificación original, lo que ha permitido buscar nuevas alternativas en función de la nueva

especificación MRML 2.0 . Actualmente se trabaja para definir nuevas funcionalidades tanto al protocolo como al sistema, entre las que destacan:

En cuanto al protocolo:

1. Soporte para aspectos de seguridad: manejo de sesiones seguras, encriptación, autenticación a nivel de servidor, cliente y mensajes, etc
2. Soporte para manejo de variables de sesión y uso de configuraciones específicas para cada usuario.
3. Soporte para la definición de Vectores Caracterísitcos y Algoritmos de Extracción de Características.
4. Mejorar las definiciones de API's para las colecciones y Repositorios de Imágenes, de manera que se puedan utilizar colecciones mediante un protocolo bien definido, y no solamente mediante el uso de URL's de cada objeto individual.

En cuanto al Sistema Huya, nuestros trabajos se dirigen a:

1. Implementar el manejo de sesiones y la especificación 2.0 (en cuanto esté oficialmente disponible)
2. Adaptar el sistema para el uso de nuevas Estructuras de Datos Multidimensionales, y nueos tipos de Objetos Multimedia como audio y video. Actualmente, algunos avances en este sentido se realizan bajo la dirección del Dr. Joshua Reiss de la universidad Queen Mary en Londres, Inglaterra **[12]**.
3. Relizar experimentación con el sistema GIFT a fin de determinar la portabilidad real de lis sistemas y del protocolo en sí.
4. Definir Repositorios de Imágenes Distribuidos mediante el uso de plataformas abiertas.

Los códigos fuente de la versión actual de Huya, así como documentación adicional, están disponibles en el sitio del proyecto Huya **[14]**

Agradecimientos

Este trabajo se ha desarrollado con la colaboración de varios estudiantes de la Licenciatura en Computación de la Universidad Central de Venezuela. La consola de administración es producto del trabajo de la Br. Marlyn Castro, y los algoritmos de Extraccción de Características usados así como las Estructuras de Datos Multidimensionales fueron programados por Milagors Rengel y Edgar Padilla **[13]** .

Referencias

[1] Eakins J., Graham M.. Content-Based Image Retrieval. *Internal Report Nr. 39, University of Northumbria at Newcastle*, October 1999.

[2] Fernández M., Simeon J., Wadler P., Cluet S., Deutsch A., Florescu D., Levy A., Maier D., McHugh J., Robie J., Suciu D., Widom J., XML query languages: Experiences and exemplars,disponible en *http://www-db.research.belllabs.com/user/simeon/xquery.ps,citeseer.ist.psu.edu/fern99xml.html*,1999

[3] The GIFT (GNU Image Finding Tool) project. http://www.gnu.org/software/gift/gift.html

[4] Goodrum A., Image Information Retrieval: An Overview of Current Research, *Journal of Informing Science, Special Issue on Information Science Research*, vol 3, number 2, 2000.

[5] Hernández Y., Diseño e Implementación de un Servidor MRML, *Trabajo Especial de Grado presentado ante la Universidad Central de Venezuela*, Escuela de Computación, Caracas, Noviembre 2003

[6] Laaksonen J., Koskela M., Laakso S., Oja E.. Self-Organizing Maps as a Relevance Feedback Technique in Content-Based Image Retrieval. *Pattern Analysis & Applications*, vol 4 number 2, pp. 140-152, June 2001.

[7] Liu P.. An Approach to specifiyng and querying multimedia objects and scheduled structures in XML *Documents. Spatial/Temporal Databases Meeting*, Paris 2000.

[8] Müller W. , Müller H., Marchand-Maillet S., et al. MRML: A communication protocol for content-based image retrieval. *International Conference on Visual Information Systems* . Lyon, France,November 2-4, 2000.

[9] Müller W., Pecenovic Z., Müller H., Marchand-Maillet S., Pun T. Et al. MRML: An Extensible Communication Protocol for Interoperability and Benchmarking of Multimedia Information Retrieval Systems. *SPIE Photonics East - Voice, Video, and Data Communications*, Boston, MA, USA, November 5--8, 2000.

[10] PicSOM image Browsing System: sitio web del proyecto PicSOM: http://www.cis.hut.fi/picsom

[11] QBIC: IBM research group. Official Web Site of the Query By Image Content of IBM, http://wwwqbic.almaden.ibm.com

[12] Reiss, Joshua: sitio web del Departamento de Electricidad de la Universidad Queen Mary, Londres,

http://www.elec.qmul.ac.uk, http://www.elec.qmul.ac.uk/~josh

[13] Rengel M., Padilla E., Estudio Comparativo entre Estructuras de Datos Multidimensionales, *Trabajo de Grado presentado ante la Universidad Central de Venezuela*, Escuela de Computación, Caracas, Octubre 2002

[14] UCV Huya: sitio web del proyecto Huya. http://strix.ciens.ucv.ve/~mrml

[15] Wang J. Z., Li J., Wiederhold G.,SIMPLIcity: Semantics-Sensitive Integrated Matching for Picture LIbraries,*IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol 3, number 9, pp 947-963, 2001.