Escuela Técnica Superior de Ingeniería

Reconocimiento de Objetos Multi-clase Basado en Descriptores de Forma

Memoria del proyecto final de carrera correspondiente a la titulación de Ingeniería Superior Informática realizado por **Andreu Hidalgo Chaparro**, dirigido por **Ágata Lapedriza** y codirigido por **Sergio Escalera** y **Xavier Baró**.

Bellaterra, 16 de junio de 2008

El firmante, Ágata Lapedriza, profesora del Departamento de Ciències de la Computació de la Universidad Autónoma de Barcelona

CERTIFICA:

Que la presente memoria ha sido realizada bajo su dirección por Andreu Hidalgo Chaparro

Bellaterra, 16 de junio de 2008

Ágata Lapedriza

Agradecimientos

Primero de todo, empezar agradeciendo a Ágata Lapedriza, directora de este proyecto, y a Sergio Escalera y Xavier Baró, codirectores de este proyecto, por la ayuda y paciencia que han tenido por aguantarme durante tanto tiempo.

Agradecer a Maximino Estévez, mi compañero de proyecto, por haber estado tanto tiempo a mi lado en estos grandes años de mi vida, no solo en el mundo universitario, sino en el laboral. Sin él las cosas hubiesen sido tan fáciles. Amigos así, son difíciles de encontrar.

Recordar a Sergio Escalera, no solo por ser mi codirector en este proyecto, sino también por los momentos que pasamos juntos en nuestros inicios en la universidad junto con Pepelu. También destacar a todos los compañeros pasados por la Carrera como Galis, Vanessa, Tania, Edgar, Gomez, Garru, Igual, Mario, los Guerrero, Santi, entre mucha más gente maravillosa que he conocido.

Gracias a todos los compañeros del Bar de la Sala de estudios del SIM, Mus, Marina, Grego, Janna, Alba, Jaime, Tati, Noe, Miguel, Irene, Nuria, Natalie, Rocio, que son unas grandes personas y con ellos los momentos en los estudios han sido muy amenos.

A los colegas del curro Oscar, Cristina, Robert, Álex y Salva, que levantarse a las 4 de la mañana, sin ellos, no sería lo mismo.

A Brad y compañía que sin ellos las fiestas no hubiesen sido fiestas.

Y sobre todo a mis padres y hermanas, que aunque estos últimos años han sido un poco difíciles, quiero que sepan que les quiero mucho. Siempre os recordaré.

Muchas gracias a todos por haber sido así de buenos conmigo. Éste proyecto os lo dedico a vosotros. De corazón.

Resumen

Este trabajo presenta un sistema para detectar y clasificar objetos binarios según la forma de éstos. En el primer paso del procedimiento, se aplica un filtrado para extraer el contorno del objeto. Con la información de los puntos de forma se obtiene un descriptor BSM con características altamente descriptivas, universales e invariantes. En la segunda fase del sistema se aprende y se clasifica la información del descriptor mediante Adaboost y Códigos Correctores de Errores. Se han usado bases de datos públicas, tanto en escala de grises como en color, para validar la implementación del sistema diseñado. Además, el sistema emplea una interfaz interactiva en la que diferentes métodos de procesamiento de imágenes pueden ser aplicados.

Aquest treball presenta un sistema per detectar i classificar objectes binaris segons la forma d'aquests. En el primer pas del procediment, s'aplica un filtrat per extreure el contorn de l'objecte. Amb la informació dels punts de forma s'obté un descriptor BSM amb característiques altament descriptives, universals i invariants. A la segona fase del sistema s'aprèn i es classifica la informació del descriptor mitjançant Adaboost i Codis Correctors d'Errors. S'han usat bases de dades públiques, tant a escala de grisos com a color, per a validar la implementació del sistema dissenyat. A més, el sistema empra una interfície interactiva a la qual diferents mètodes de processament d'imatges poden ser aplicats.

This paper presents a system to detect and classify binary objects according to their shape apparence. In the first step of procedure, a filter is applied to extract the contour of the object. With the information points, a descriptor BSM is obtained with highly descriptive, universal and invariant features. At the second step of the system, tehe description is learnt and classified through Adaboost and Error Correction Codes. We have used public databases, on both, greyscale and color images to validate the implementation of the designed system. In addition, the system uses an interactive interface in which different image processing methods can be applied.

Key words: Inteligencia Artificial, Visión por computador, Detección de objetos, Descripción de objetos, Clasificación multi-clase.

Índice

Índice de figuras

1. Introducción

Con el paso del tiempo el hombre ha necesitado obtener ciertos parámetros visuales sobre objetos que le permitan controlar de forma automática determinadas situaciones. Situaciones en las que el hombre, ya sea por comodidad o necesidad, se ayuda de la computación para problemas cotidianos y no tan cotidianos de la vida real. Los seres humanos como la gran mayoría de animales alimentan sus cerebros con sensaciones, estímulos, percepciones para la realización de una acción determinada. Uno de los sentidos más importantes, sin despreciar los demás, es la vista, estímulos visuales representados en el cerebro que nos permiten observar y reconocer objetos distantes sin necesidad de tener contacto con ellos. La Visión Artificial es una tarea increíblemente difícil, una tarea que parece relativamente trivial para los humanos y que es infinitamente compleja de llevar a cabo para las computadoras.

La detección de objetos es un trabajo de Visión por Computador que lleva cerca de 50 años en investigación, sobre el que se ha invertido gran esfuerzo debido a su amplitud de posibilidades prácticas. La detección automática de objetos en imágenes permite un avance sobre la Visión Artificial, facilitando la interacción de los robots con su entorno. El campo de la Visión por Computador tiene como fin extraer propiedades del mundo a partir de un conjunto de imágenes. Permite mediante la adquisición por CCD, dispositivo de cargas eléctricas interconectadas, obtener una imagen digital de un objeto real del cual podremos analizar ciertos parámetros mediante un sistema informático de manera que se pueda implementar un algoritmo que nos permitirá resolver un problema concreto. En la actualidad, cada vez se utiliza más la ayuda de la Visión por Computación y se están desarrollando diferentes métodos orientados a diversas temáticas, como pueden ser la industria, seguridad, medicina, inspección, telemetría, robótica, etc.

En aplicaciones de robótica, el robot continuamente busca objetos en imágenes adquiridas mediante cámaras. Estos objetos capturados en imágenes digitales serán procesados mediante algoritmos de reconocimiento y aprendizaje, para extraer la máxima información y analizar el entorno que le rodea. En una segunda fase, se realizaría la clasificación de la información procesada por medio de patrones singulares y clasificadores, almacenando la información extraída de los objetos para realizar posteriores comparaciones contra nuevos objetos. Estas imágenes son automáticamente etiquetadas con descripciones simbólicas que pueden semejarse a determinados objetos en una base de datos. Muchos son los algoritmos realizados para el reconocimiento e interpretación de objetos, según los parámetros que se quiera observar, forma, color, contorno, volumen, etc, se utilizará uno u otro. El reconocimiento puede ser también útil para aplicaciones comerciales. Recientemente, se han desarrollado sistemas web para la búsqueda de objetos en Internet [5].

El reconocimiento por forma o contorno de objetos es una de las técnicas más populares del Reconocimiento de Patrones. Su objetivo consiste en resolver el problema del modelado y reconocimiento de objetos procedentes de un amplio conjunto de clases. Es una tarea sumamente difícil debido a la alta variabilidad de la apariencia del los objetos: cambios en la perspectiva, diferentes puntos de vista, oclusiones, deformaciones rígidas y elásticas, etc. Un gran esfuerzo se ha hecho en el última década para desarrollar un buen detector de características y contornos inspirados en enfoques estructurales o estadísticos de patrones de reconocimiento [6]. En general, se pueden ver dos puntos de interés: la definición de los descriptores de forma compacta, y la formulación de métodos sólidos de clasificación de acuerdo a los descriptores.

Los trabajos que encontramos en la literatura sobre el reconocimiento de objetos tienen como punto común de partida la extracción de características altamente descriptivas de los objetos a categorizar. La extracción de un conjunto rico de características será fundamental para obtener un aprendizaje robusto y una clasificación exitosa. En este punto, nuestro objetivo será comparar diferentes descriptores usados en la literatura. Se pretende analizar las ventajas y desventajas del estado del arte para detectar las necesidades de los descriptores cuando el conjunto de objetos entre los cuales queremos distinguir es elevado. Para ello también se pretende realizar modificaciones y presentar nuevos descriptores que consigan superar los resultados obtenidos hasta el momento.

El siguiente paso a realizar se centra en la clasificación de objetos. Una vez se han descrito los diferentes objetos a partir de unas características altamente descriptivas, universales e invariantes, tenemos que usar una técnica de aprendizaje que nos seleccione los datos más relevantes para conseguir hacer predicciones exitosas. En este punto nos centraremos en el aprendizaje estadístico para comparar las diferentes técnicas más usadas por los autores. Básicamente nos centraremos en Adaboost [1][2], que es una técnica de agrupación de varios clasificadores, cuyo algoritmo tiene una gran capacidad de selección y de alto rendimiento cuando es aplicado en problemas binarios. Cabe destacar que estas técnicas suelen obtener resultados exitosos en problemas binarios, es decir, cuando aprendemos a distinguir entre únicamente dos grupos de objetos. Para extender este comportamiento binario a multi-clase (distinguir entre un número elevado de clases candidatas), nos centraremos en los Códigos Correctores de Errores (Error-Correcting Output Codes), que han demostrado en diferentes aplicaciones reales extender de forma satisfactoria clasificadores binarios a multi-clase [4].

El objetivo principal de este proyecto consiste en clasificar de forma automática objetos en imágenes de entre un conjunto elevado de clases candidatas mediante técnicas de Visión por Computador. Tal proceso se realiza mediante la identificación y extracción de características de contornos rele-

vantes, que pueden ser usadas como entradas a un sistema de categorización multi-clase. Por un lado, introducimos el Blurred Shape Model (BSM), que codifica la probabilidad espacial de aparición de la forma de un objeto y su información contextual. Como resultado se obtiene una sólida técnica frente a deformaciones elásticas. Por otro parte, se presenta un esquema exitoso para describir y clasificar objetos binarios basados en ECOC.

Blurred Shape Model esta basado en el reconocimiento de formas y contornos de objetos. Con este método se pretende obtener un descriptor del contorno de un objeto, el cual es comparado en una base de datos de imágenes y descriptores aprendidos previamente. Estos resultados obtenidos de aplicar el sistema BSM serán añadidos en la base de datos, siendo ésta más completa cada vez que se aplique el método. Este sistema se compone de un descriptor y un clasificador, el cual usa la información de los descriptores para obtener una clase candidata semejante al objeto mediante métodos de Adaboost. Además de realizar el sistema multi-clase, se ha realizado una interfaz, ORMGPlayer, para trabajar con proyectos de Visión por Computador e Inteligencia Artificial de forma amigable, y que será disponible para que futuros proyectistas puedan testear sus algoritmos implementados. Esta aplicación multimedia está diseñada para la visualización de los resultados de los métodos de detección y reconocimiento de objetos, características faciales, puntos de interés, etc. implementados en este proyecto. Finalmente se mostrará la robustez de este sistema y su uso sobre diferentes conjuntos de objetos multi-clase, tanto procedentes de datos sintéticos como de entornos reales.

Esta memoria está dividida de la siguiente manera: el capítulo 2, 3 y 4 abarca la descripción del Sistema creado para este proyecto, detallando cómo se obtiene un descriptor Blurred Shape Model mediante filtrados de imagen, el aprendizaje por Boosting de la información obtenida de los descriptores y clasificación de estos por ECOC. El capítulo 5 presenta la interfaz diseñada para la visualización de los resultados adquiridos de aplicar nuestro sistema u otros métodos implementados por otros programadores. En el capítulo 6 se combinan todos los métodos para el diseño del sistema BSM. Las pruebas y experimentos de la aplicación ORMGPlayer y el sistema BSM se realiza en el capítulo 7, y finalmente terminaremos con las conclusiones, anexos y contenido del CD.

2. Descriptor BSM

Blurred Shape Model se basa en descriptores por forma de objetos, lo que permite la definición espacial de las regiones en las que algunas partes de la forma pueden estar implicadas. En la Tabla 1 se muestra el algoritmo BSM.

Tabla 1. Algoritmo BSM.

Dada una imagen binaria I,
Obtener la $forma\ S$ contenida en I
Dividir I en $n \times n$ subregiones de tamaños iguales $R = \{R_1, \ldots, R_{n\times n}\}$
Dividida I $n\times n$ subregiones de tamaño igual $R = (r_i, \ldots, n\times n)$, con c_i el centro de coordenadas de cada región r_i, $N(r_i)$ es el vecino de la región regiones r_i, que se define como $N(r_i) = (r_k | r \in R, ||c_k - c_i||^2 \leq 2 \times g^2)$, donde g es el tamaño de la celda.

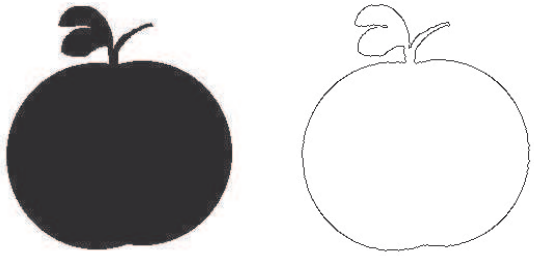
Para cada punto $x \in S$,
  Para cada $r_i \in N(rx)$,
    $d_i = d(x, r_i) = ||x - c_i||^2$
  Fin Para
  Actualizar las probabilidades de las posiciones del vector v como:
    $v(r_i) = v(r_i) + \frac{1/d_i}{D_i}$, $D_i = \sum_{ck \in N(r_i)} \frac{1}{||(x-ck)||^2}$
Fin Para

Normalizar el vector v como: $v = \frac{v(_i)}{\sum_{j=1}^{n^2} v(_j)}\ \forall i \in [1, \ldots, n^2]$

Dada una imagen binaria, se le aplica un filtro digital llamado Canny [17]. Este filtro nos retorna una imagen de contorno del objeto, como se muestra en la Fig. 1 (b). A partir de está imagen se extrae la información de la forma del objeto, y con ello el descriptor BSM. Canny es uno de los métodos más importantes para realizar una detección global de bordes sobre una imagen por su comportamiento con el tratamiento digital ante ruido. Esta técnica, que se caracteriza por estar optimizada para la detección de bordes diferenciales, consta de tres etapas principales: filtrado, decisión inicial e histéresis [13].

A continuación, realizado el filtrado digital, la imagen resultante es dividida en $n \times n$ regiones. Dado un conjunto de puntos de contorno de un objeto, éstos serán utilizados para el cálculo del descriptor BSM. La imagen de la región se divide en una cuadrícula de $n \times n$, donde n es el tamaño de la subregión (el tamaño de la red identifica el nivel de resolución de difuminación del objeto). En la Fig. 3 se puede observar los resultados obtenidos al aplicarle diferentes

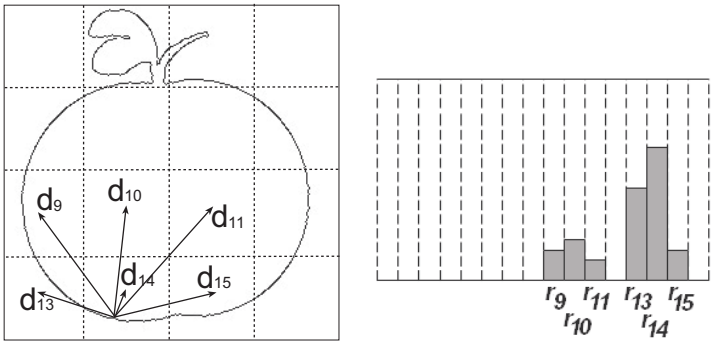
valores al tamaño de la cuadrícula. Cada región recibe unos valores de los puntos de contorno de la región y de los puntos de sus vecinos. De este modo, cada punto de contorno contribuye a medir la densidad de su región y la de sus vecinos. Esta contribución se pondera de acuerdo a la distancia entre el punto y el centro de coordenadas c_i y la región R_i.



(a) (b)

Figura 1. (a) Imagen de muestra (b) Imagen contorno

Para dar la misma importancia a cada punto de contorno, todas las distancias a los centros de los vecinos se normalizan. El descriptor de salida es un vector histograma v de longitud $n \times n$, donde cada posición se corresponde con la distribución espacial de los puntos de contorno en el contexto de la región que la contiene y a sus vecinos. En la Fig.2(a) se muestran las distancias de un punto de forma a sus regiones vecinas, y en la Fig.2(b) se puede ver un histograma de densidad de los valores del punto de forma.



(a) (b)

Figura 2. Ejemplo BSM de la estimación de densidad

El histograma resultante, obtenido de procesar todos los puntos de contorno, es normalizado en el rango $[0 \ldots 1]$ para obtener la función de densidad de probabilidades de $n \times n$ bins. De esta manera, el descriptor de salida representa una distribución de probabilidades del contorno del objeto en una distorsión espacial, donde el nivel de la distorsión se determina por el tamaño de la red. En cuanto la complejidad computacional, para una región de $n \times n$ píxeles,

siendo k el numero de puntos de contorno de la imagen requiere un coste de $O(k)$ operaciones simples.

(a) (b) (c)

Figura 3. (a) 10×10 BSM (b) 20×20 BSM (c) 32×32 BSM

3. Adaboost

La agrupación de varios clasificadores, obtenidos utilizando un mismo método, es una manera natural de incrementar la precisión, con respecto a la obtenida con la utilización aislada de dichos clasificadores. Uno de los métodos más populares para crear estas agrupaciones de clasificadores es el boosting. Este término engloba toda una familia de métodos, de la que AdaBoost es la variante más conocida. Estos métodos trabajan asignando un peso a cada ejemplo. Inicialmente, todos los ejemplos tienen el mismo peso. En cada iteración, se construye un clasificador, denominado base o débil, utilizando algún método de aprendizaje, y teniendo en cuenta la distribución de pesos. A continuación, el peso de cada ejemplo se reajusta, en función de si el clasificador base le asigna la clase correcta o no. El resultado final se obtiene mediante voto ponderado de los clasificadores base.

Tabla 2. Algoritmo Adaboost

Empezar con pesos $w_i = 1/N$, $i = 1, 2, \ldots, N$, $F(x) = 0$.
Repita el procedimiento para $m = 1, 2, \ldots, M$:
Ajustar la función de regresión $f_m(x)$ con mínimos cuadrados ponderados de y_i hacia x_i con pesos w_i.
Actualizar $F(x) \leftarrow F(x) + f_m(x)$
Actualizar $w_i \leftarrow w_i e^{-y_i f_m(x_i)}$ y normalizar.
Clasificador de salida $sign[F(x)] = sign[\sum_{m=1}^{M} f_m(x)]$

La versión original de AdaBoost trabaja sólo con problemas binarios. Por esta razón, Adaboost [1] ha sido elegido para impulsar el BSM de diferentes clases modelo con el fin de definir una clasificación basada en las características que mejor discriminan una clase contra otra. El algoritmo del método Adaboost se puede ver en la Tabla 2. Cuando los objetos son muy similares, solo con ligeras deformaciones en partes compartidas pueden incluir distancias de errores importantes que finalmente podrían llevar a una perdida de la clasificación de los objetos. Observar la Fig. 4. Los dos objetos tienen una región discriminativa que divide las dos categorías (marcado con un círculo). Adaboost se centra en estas regiones mediante la selección de las característica más discriminativa entre objetos.

Figura 4. Regiones discriminantes entre objetos.

4. ECOC: Códigos de Corrección de Errores

El análisis de la evolución de ECOC ha demostrado que corrige los errores causados por los algoritmos de aprendizaje. Si un punto de contorno es incorrectamente clasificado por algunas de las dicotomías aprendidas, todavía puede ser clasificado correctamente después de haber sido decodificado debido a la capacidad de corrección del algoritmo. ECOC es un marco de trabajo sencillo, pero poderoso, para hacer frente a los problemas de categorización multi-clase a partir de la combinación de clasificadores binarios.

Dado un conjunto de N clases, la base ECOC consiste en el diseño de un elemento clave en cada una de las clases. Almacenando estos elementos en una matriz M por filas, se define M como, $M \in \{-1, 0, 1\}^{N \times n}$, siendo n la longitud del código. Desde el punto de vista de aprendizaje, M se construye considerando n problemas binarios, cada uno correspondiente a una columna de la matriz. Agregando clases en grupos, cada división define una partición de clases (codificados con +1, -1 dependiendo de su clase miembro, ó 0 si la clase no está considerada parte del problema binario). En la Fig. 5 se puede ver un problema binario entre tres clases con agrupaciones de uno-contra-uno.

(a) (b) (c)

Figura 5. Problema binario entre 3 clases.

4.1. Codificación Binaria

La codificación binaria estándar está basada en estrategias de uno-contra-uno [15], estrategias de uno-contra-todos [14] y la estrategia aleatoria compacta [16]. Una vez definidas las particiones de las clases, en la estrategia uno-contra-uno, cada una de ellas se codifica como un columna de la matriz M de codificación, como se muestra en la Fig. 6 (c). Las regiones oscuras son codificadas como 1 (primera división de clases), y el gris, regiones son codificados como -1 (segunda división de clases). Las regiones blancas corresponden a las clases no consideradas por sus respectivos clasificadores. Ahora, las filas de la matriz M define los elementos claves (Y_1, Y_2, Y_3) para sus clases corre-

spondientes (c_1, c_2, c_3). En uno-contra-todos, cada clasificador (ó dicotomía) está entrenado para distinguir un clase del resto de clases. Dados N clases, esta técnica tiene una longitud código de N bits. Un ejemplo de la estrategia ECOC de uno-contra para un problema de tres clases se muestra en la Fig. 6 (a). La estrategia aleatoria de densidades genera un gran número de matrices M de codificación aleatoria de longitud n, dónde los valores {+1,-1} tienen la misma probabilidad que aparezcan ($P(1) = P(-1) = 0{,}5$). Estudios sobre el comportamiento de la estrategia ECOC aleatoria compacta sugiere una longitud de $n = 10logN$ [16]. Un ejemplo de un diseño de la estrategia aleatoria de densidades para un problema de tres clases se muestra en la Fig. 6 (b). En la Fig. 6 (c) se muestra, para el mismo problema de tres clases, la estrategia uno-contra-uno, que es la utilizada en este proyecto como estrategia ECOC.

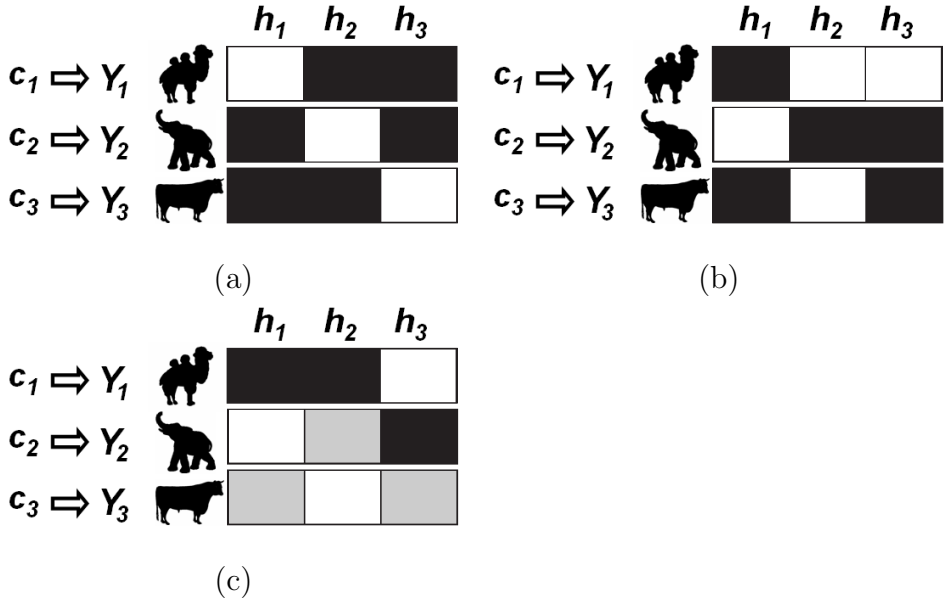


Figura 6. Codificación ECOC. (a) uno-contra-todos. (b) aleatoria de densidades (c) uno-contra-uno

4.2. Decodificación binaria

En el paso de decodificación, aplicando los n clasificadores binarios entrenados, se obtiene un código para cada punto de datos en el test. Este código se compara con la base de elementos clave $(y_i, i \in [1, .., N])$ de cada clase que se define en la matriz M, y el punto de datos se le asigna a la clase con el elemento "más cercano". Los diseños de decodificación binaria más utilizados son: Decodificación de Hamming [22], Decodificación Inversa de Hamming [23], y Decodificación Euclidana [24].

- *Decodificación de Hamming*

Esta estrategia de decodificación se basa en los principios de corrección de errores, suponiendo que la tarea de aprendizaje puede ser modelada como un problema de comunicación, donde la información de la clase se transmite sobre un canal, obteniendo dos posibles símbolos en cada posición de la secuencia [?]:

$$HD(x, y_i) = \sum_{j=1}^{n}(1 - sign(x^j y_i^j))/2$$

- *Decodificación Inversa de Hamming*

La decodificación inversa de Hamming viene definida como: siendo Δ la matriz compuesta por la decodificación de Hamming mediante los códigos de M. Cada posición de Δ está definida por $\Delta(i_1, i_2) = HD(y_{i_1}, y_{i_2})$. Δ puede ser invertida para hallar el vector que contiene las N clases individuales, que viene definido por:

$$IHD(x, y_i) = max(\Delta^{-1} D^T)$$

dónde los valores de $\Delta^{-1} D^T$ pueden verse como la proporcionalidad de cada clase respecto al código de test, y D es el vector de valores de decodificación de Hamming de la prueba del elemento del test x para cada uno de los códigos base y_i.

- *Decodificación Euclidiana*

Otra buena estrategia de decodificación binaria es la Decodificación Euclidiana.

$$ED(x, y_i) = \sqrt{\sum_{j=1}^{n}(x^j y_i^j)^2}$$

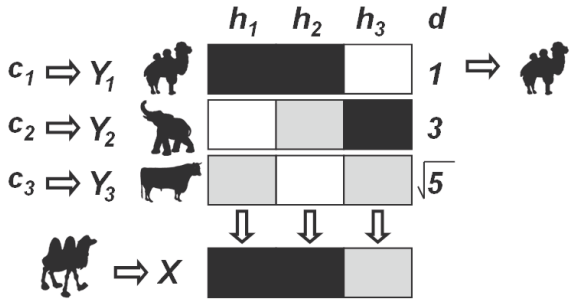


Figura 7. DECOC. Decodificación Euclidiana

En la Fig. 7 se muestra una decodificación binaria por esta estrategia para el problema binaria de tres clases de la Fig. 5. Esta decodificación ECOC suele dar mejores resultados que las anteriores, por esa razón es la que utilizamos en este proyecto.

5. ORMGPlayer

La interficie, llamada ORMGPlayer, es una aplicación multimedia diseñada para la ejecución y visualización de los resultados de los algoritmos implementados para la detección de objetos, pero también podemos utilizar dicha aplicación para reproducir cualquier objeto multimedia: listas de fotos, pistas de audio o vídeo, entre otras.

Los formatos que ORMGPlayer reconoce son, para archivos de imágenes, las extensiones jpg, jpeg, bmp y gif, para archivos de video, avi, mpg , mpeg y wmv y finalmente para archivos de audio, mp3, mid, midi, entre otros. No todos estos objetos multimedia podrán ser utilizados por todos los métodos, ya que el formato a utilizar viene determinado por los algoritmo implementado por el programador.

El reproductor está implementado mediante librerías MFC (Microsoft Fundation Clases), que nos permiten realizar el diseño de una interfaz. La biblioteca MFC es un marco de trabajo de aplicaciones para la programación en Microsoft Windows. Desarrollada en C++, MFC proporciona gran parte del código necesario para administrar ventanas, menús y cuadros de diálogo, etc.

El marco de trabajo MFC es una poderosa herramienta que permite aprovechar el código desarrollado por programadores expertos en Windows. MFC reduce el tiempo de desarrollo; hace el código más portable a otras plataformas; proporciona un elevado grado de compatibilidad sin sacrificar la libertad y la flexibilidad de programación; y otorga fácil acceso a los elementos de la interfaz de usuario y tecnologías "difíciles de programar", como la tecnología Active, OLE y la programación para Internet. Más aún, MFC simplifica la programación con bases de datos a través de Data Access Objects (DAO) y la Conectividad abierta de bases de datos (ODBC), así como la programación para redes con Windows Sockets. ORMGPlayer utiliza DirectShow, uno de los componentes de la tecnología DirectX, que nos permitirá la visualización y reproducción de objetos multimedia.

Microsoft DirectX es una colección avanzada de interfaces de programación de aplicaciones (APIs) integrada a los sistemas operativos Microsoft Windows. Este conjunto de APIs mantiene una plataforma de desarrollo de aplicaciones multimedia estándar para aplicaciones Windows, permitiéndoles a los programadores del software acceder al hardware especializado sin tener que escribir código específico de cada tipo de hardware.

Cuando una aplicación o un juego es escrito para DirectX, el programador no tiene que preocuparse por exactamente qué tarjeta de sonido o por el adaptador gráfico que el usuario final podría tener en su máquina, DirectX se encarga de eso para ello. DirectX juega un papel en muchas funciones,

incluyendo renderización 3D, reproducción de video, interfaces para joysticks y ratones, gestión de redes para multi-jugador y muchos más. Sin embargo, DirectX tiene una gran desventaja: no es portable, es decir, una aplicación programada con DirectX está condenada a trabajar solamente en Windows, lo cual no es nada deseable a menos que sepas que el único mercado al que va dirigido la aplicación son personas con una computadora con sistema operativo Windows.

DirectShow de Microsoft es una arquitectura multimedia y una increíble mejora frente a la anterior conocida como la interfaz de control de soporte (Media Control Interface, MCI). Debido a las limitaciones inherentes de una MCI de 16 bits, como la necesidad de controladores de única función hinchados, se diseñó DirectShow para alojar la amplia selección de hardware y tecnologías multimedia actuales que el anterior no podía. Basado en Component Object Model (COM) de Microsoft, los principales problemas de MCI que consistían en interfaces inconsistentes es ahora un problema olvidado cuando se utiliza DirectShow con sus características de multiproceso y multitarea. DirectX, un conjunto de interfaces de programación de aplicación a bajo nivel (API) para la creación de aplicaciones multimedia de alto rendimiento con su principal objetivo de diseño acelerado, se diseñó originalmente para mejorar la plataforma de juegos de Windows 95. Pero con la llegada del DVD, DBS y una fuente de nuevas tecnologías, actualmente se utiliza como la pasarela para acceder a diferentes periféricos hardware y como parte integrante del sistema operativo Windows OS (ediciones 98 y ME) y de Windows NT 5.0.

5.1. Aplicación

ORMGPlayer es una aplicación con un diseño y una funcionalidad muy sencilla como se puede observar en la Fig. 8. Este reproductor de compone de:

- Ventana de reproducción donde el usuario podrá visualizar o escuchar los objetos multimedia seleccionados.
- Ventana de reproducción resultado de aplicar un método, ya sea FaceDetect, CircleCercle o BSM, a un objeto multimedia.
- Botones de reproducción para el manejo de los archivos.
- Un cuadro de selección de métodos, el cual nos permite elegir el algoritmo que se quiera aplicar a dicho objeto.
- Botón Options, abre un cuadro de opciones del método seleccionado.
- Botones de tratamiento de ficheros, para la apertura de objetos multimedia y guardado de los objetos resultantes de aplicarle un método.
- Una lista de reproducción donde se almacenan temporalmente los archivos abiertos por el usuario.
- Un cuadro de información donde se podrá observar en todo momento el

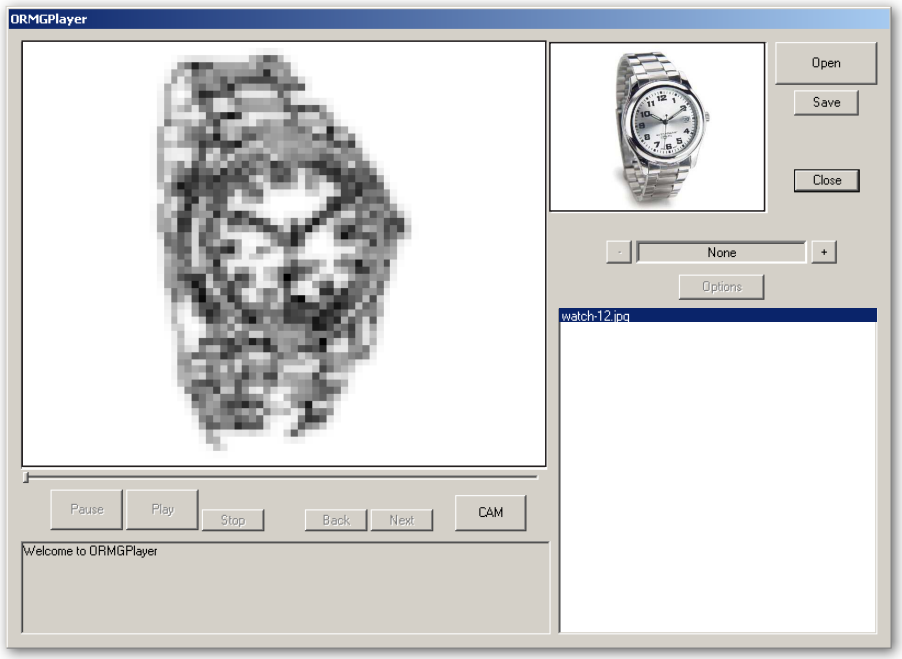


Figura 8. ORMGPlayer + BSM

objeto multimedia utilizado, el algoritmo seleccionado y el proceso en el que se encuentra dicho algoritmo.

5.2. Diagrama de Clases

La estructura de clases del reproductor se puede apreciar en la Fig. 9. Se puede ver la relación entre las funcionalidades que forman esta parte del sistema.

La clase CORMGPlayerDlg se ha programado mediante librerías MFC, y tiene herencia a la clase Dialog para la creación de una interfaz. Incluye librerías DirectX para la inicialización y reproducción de las imágenes. Esta clase se encarga de gestionar las ventanas de reproducción, como también de llamar a los constructores de las otras clases. El alcance de la clase CORMGPlayerDlg se puede ver en la Fig. 10.

La clase CControl es la responsable de controlar en todo momento el estado de los botones, siendo habilitados o deshabilitados según el algoritmo que utilicemos o el tipo de ejecución aplicado en la interfaz. Cada botón hará una llamada a los métodos correspondientes de la clase CControl. Además, almacena los datos de los elementos de la lista de reproducción en una estructura y proporciona el control de la reproducción, según el algoritmo que se aplique. Por último, muestra mensajes de información de los procedimientos ejecutados

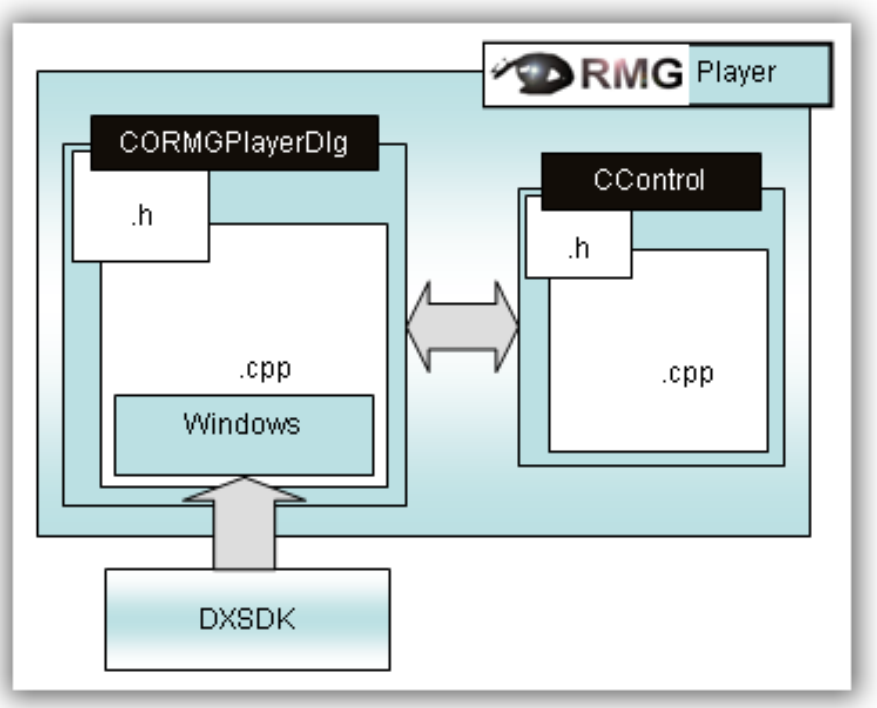


Figura 9. Diagrama de clases

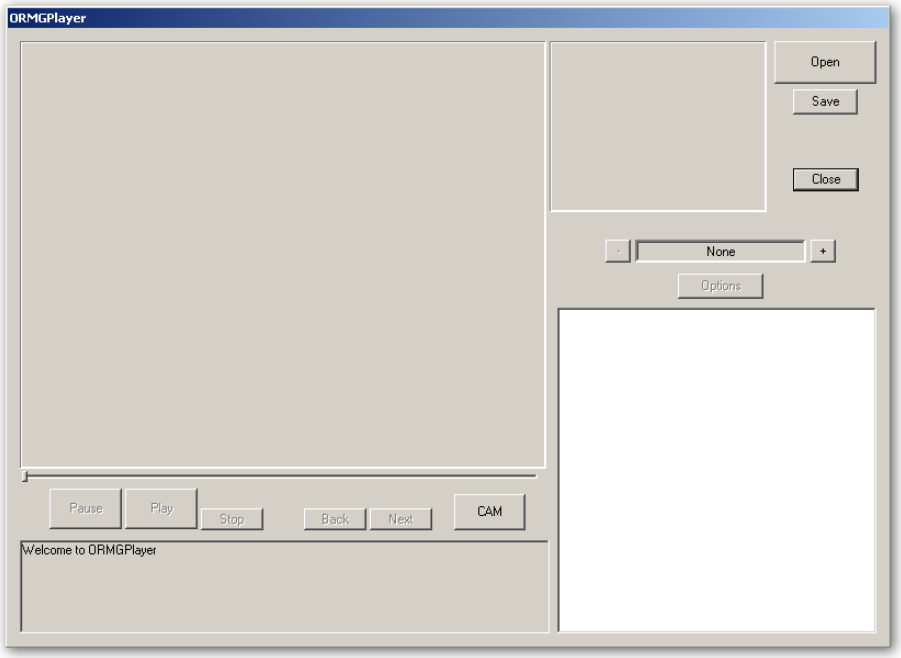


Figura 10. Ámbito de la clase CORMGPlayerDlg

en la ventana pertinente. El alcance de la clase se puede observar en la Fig. 11.

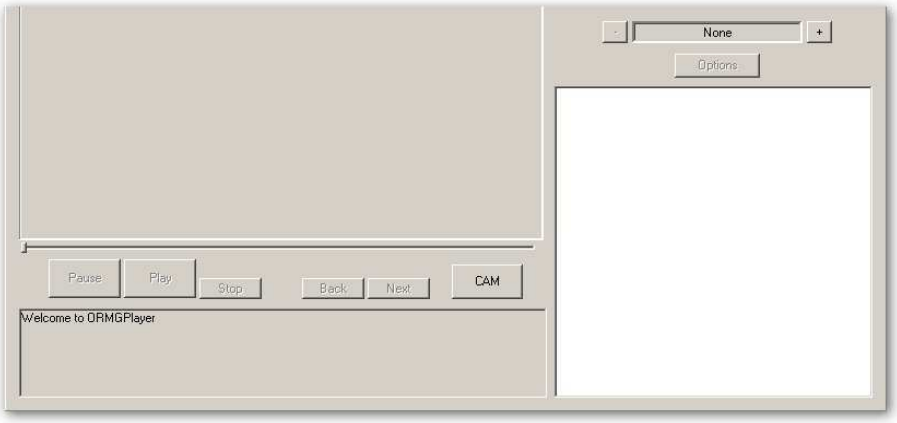


Figura 11. Ámbito de la clase CControl

5.3. Uso de la interficie

Una vez introducidas las librerías necesarias para el diseño de la aplicación y las partes que la componen, pasaremos a explicar todos los componentes que forman la interfície.

Por un lado, tenemos las dos ventanas de reproducción. La diferencia entre ellas, es que en la primera vamos a visualizar el objeto multimedia (foto, vídeos, etc.) original, y en la segunda el resultado de aplicar los métodos, Facedetect, CircleCercle y BSM que podremos seleccionar en el cuadro de algoritmos. Si no queremos utilizar ninguno de los métodos implementados, esta aplicación funcionará como un reproductor de fotos, audio o vídeo convencional. Para aplicar alguno de los métodos empezaremos por abrir el objeto u objetos multimedia, mediante el botón Open. Estos objetos aparecerán en la lista de reproducción, en la que podremos seleccionar el archivo activo. Este proceso se verá reflejado en la ventana de información, donde podremos leer el nombre del objeto que estamos seleccionando.

Una vez tenemos el objeto activo, podremos seleccionar el método a aplicar en la lista de algoritmos implementados. Al tener el método seleccionado veremos que en el cuadro de parámetros se habilitarán los propios de dicho método, con valores por defecto los cuales podremos cambiar por los que deseemos. Una vez seleccionadas las propiedades que deseemos, deberemos pulsar el botón play que se encuentra en la zona de botones de reproducción. En este momento, podremos ver en la ventana de información que ha empezado a aplicarse el método que habíamos seleccionado, y paso a paso en que punto del procesado se encuentra dicho procedimiento.

Al finalizar dicho método, aparecerá la imagen resultado en la ventana de reproducción. En este punto podremos guardar el objeto multimedia resultante de aplicar el algoritmo mediante el botón Save. Si tenemos más objetos en la

lista de reproducción podremos seleccionar otro mediante clics de ratón o mediante los botones anterior o siguiente que desplazan el foco en la lista para seleccionar otra imagen. En caso de que no queramos aplicar alguno de los métodos a otro objeto multimedia, podremos cerrar la aplicación mediante el botón de cerrado.

6. Sistema BSM

En este apartado, se presenta la integración de los diferentes módulos presentados en los capítulos anteriores, y que forman la aplicación de reconocimiento de objetos basados en forma. En la Fig. 12 se muestra el esquema de integración de los módulos. Primero, a un objeto se le extrae la imagen contorno, para seguidamente calcular su descriptor BSM. A continuación, se usa Adaboost para entrenar los clasificadores binarios para las diferentes clases y finalmente con la utilización de la herramienta de Códigos Correctores de Errores se extiende la clasificación binaria al caso multi-clase. Cuando se quiere clasificar un nuevo objeto, el proceso se repite, pero aplicando los clasificadores aprendidos previamente para obtener una decisión de clasificación.

Figura 12. Esquema del sistema

Como podemos ver en el diagrama de casos de uso, Fig. 13, existen varias líneas de ejecución en nuestro sistema. Tenemos tres casos independientes: el tratamiento multimedia, la ejecución de Face Detect y la ejecución del sistema BSM. En el caso del tratamiento multimedia, podríamos reproducir en la ventana principal cualquier tipo de vídeo, mostrar todo tipo de imágenes e incluso reproducir pistas de audio. Si el usuario escogiera la opción de Face Detect y seleccionase cualquier tipo de objeto multimedia, a excepción de pistas de audio, veríamos los resultados de este método en la ventana de resultados. Y por último, en el caso del sistema BSM visualizaríamos el objeto original en la ventana principal y en la secundaria el resultado de la clasificación multi-clase.

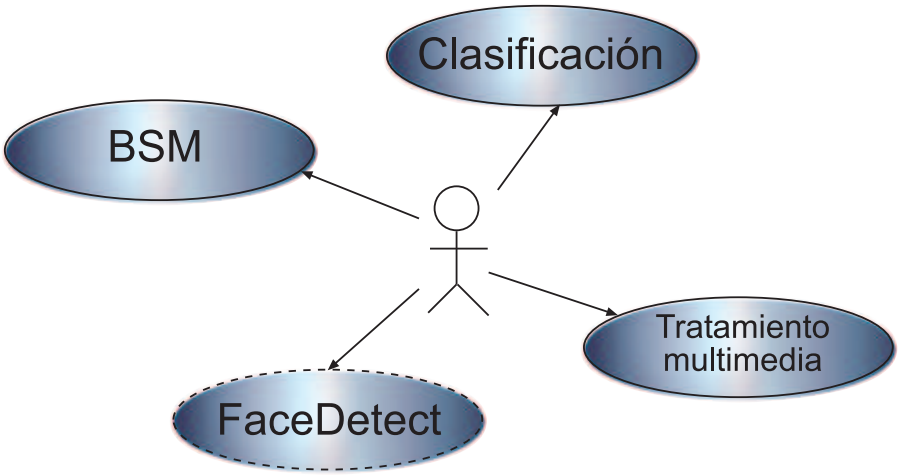


Figura 13. Diagrama de casos de uso.

En la Fig. 15 se puede observar el diagrama de clases de todo el sistema implementado en este proyecto. En dicho diagrama, podemos observar por un lado las clases CORMGPlayerDlg y CControl, clases que se utilizan para controlar el reproductor que se ayuda de las librerías de DirectShow para poder visualizar cualquier archivo multimedia. Si el usuario decide realizar la clasificación de un objeto cualquiera, se hará mediante la clase CBSM que con la ayuda de las librerías IPL y OpenCV se obtendrá una mejor implementación para dicho sistema, ya que las librerías IPL nos darán soporte para trabajar con imágenes, y las OpenCV para tratarlas. De esta manera obtendremos la una buena clasificación para este objeto, comparando su descriptor con los descriptores de todos los objetos existentes en las bases de datos MPEG07, para imágenes binarias y Caltech101, para imágenes en escala de grise y color.

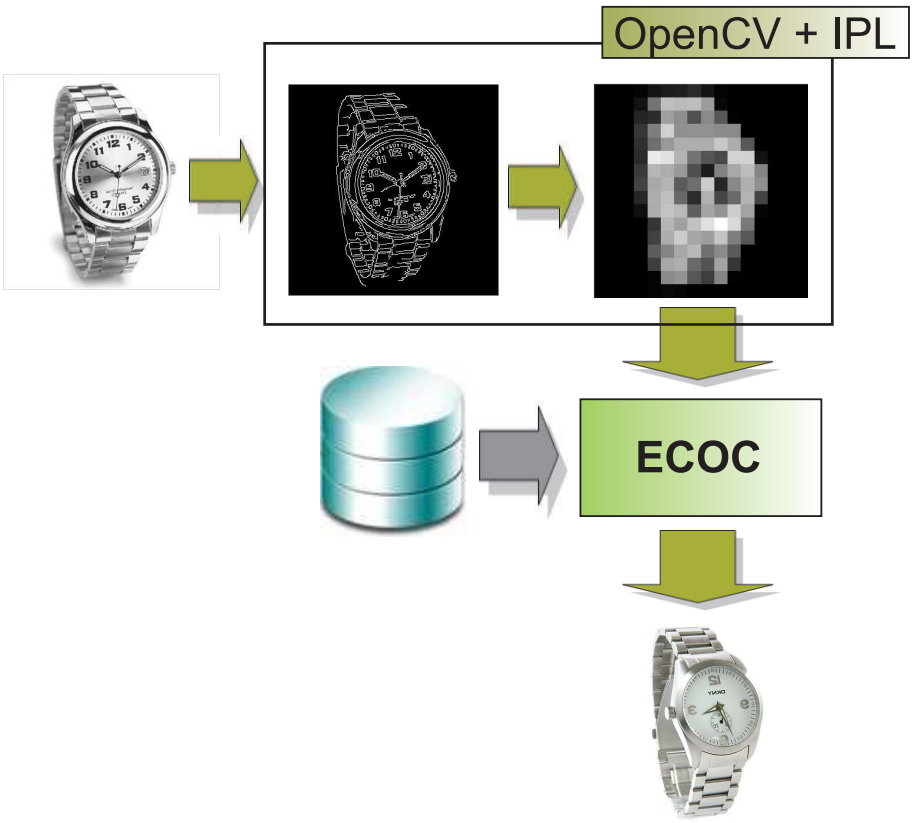


Figura 14. Esquema de funcionamiento del sistema

Por último, en la Fig. 14 podemos ver de forma gráfica el funcionamiento del proceso de clasificación. En este proceso trataremos la imagen inicial con las librerías OpenCV e IPL, obteniendo primero la imagen de contorno calculada mediante el método Canny con un threshold de 300; para después obtener su

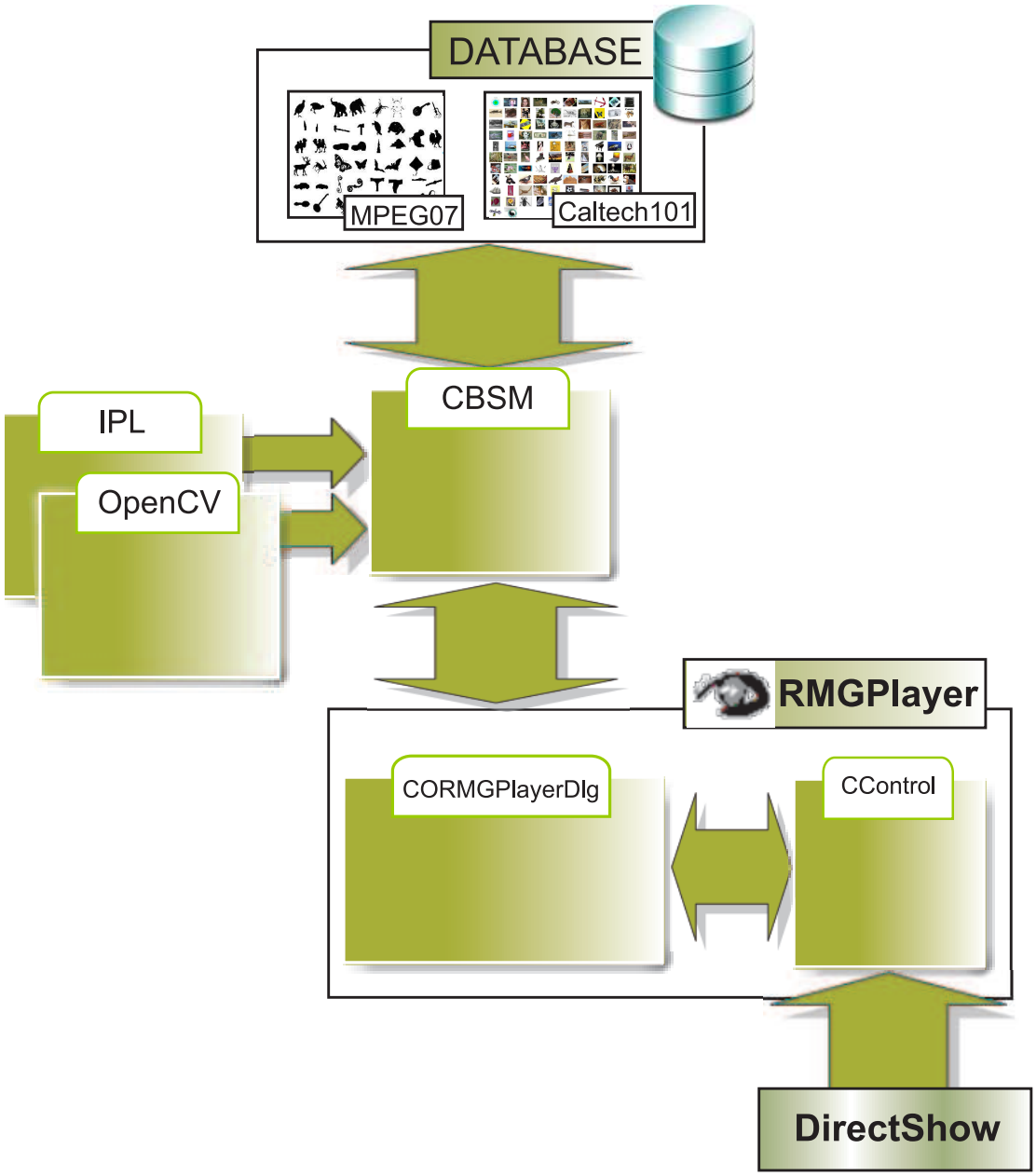


Figura 15. Diagrama de clases del sistema

descriptor BSM y compararlo con el resto de descriptores de la base de datos mediante el algoritmo ECOC con una estrategia de codificación uno-contra-uno y decodificación Euclidiana.

7. Resultados

Para la comprobación y validación del sistema de detección, aprendizaje y clasificación de objetos se procederá a explicar y cuantificar los resultados. Dividiremos está sección en tres partes: datos, métodos y experimentos.

7.1. Datos

Para testear el sistema usaremos 23 categorías del repertorio de clases de la base de datos MPEG [7]. Esta base de datos ha sido elegida ya que proporciona un alta variabilidad intra-clase en términos de escala, rotación, deformaciones rígidas y elásticas, así como una baja variabilidad inter-clase. Un par de ejemplos de cada categoría se pueden ver en la Fig. 16. Cada una de las clases contiene 20 muestras, representando un total de 460 objetos.

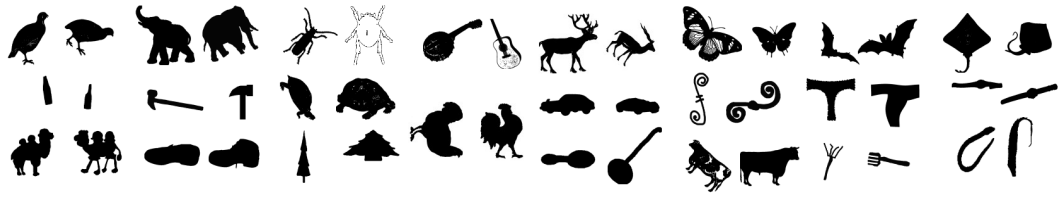


Figura 16. Clases MPEG07

Para realizar pruebas tanto en escala de grises como en color utilizamos la base de datos Caltech 101 [12]. La base de datos Caltech 101 contiene 101 categorías de imágenes de objetos. Una muestra de estas categorías se puede observar en la Fig. 17. Tiene entre 40 y 800 imágenes por categoría, pero la gran mayoría están alrededor de 50 imágenes.

7.2. Métodos y herramientas

Los parámetros utilizados para el cálculo del descriptor BSM en el sistema es de una resolución borrosa de 16×16, como se verá en los experimentos. Además, para analizar el rendimiento del sistema, el descriptor BSM es entrenado usando 50 iteraciones de Gentle Adaboost [1], y la estrategia utilizada para los Códigos de Corrección de Errores es uno-contra-uno para la codificación y la distancia Euclidiana para la decodificación ECOC.

La aplicación ORMGPlayer es la encargada de proporcionarnos la interfície para visualizar los objetos procesados por los sistemas implementados. Para

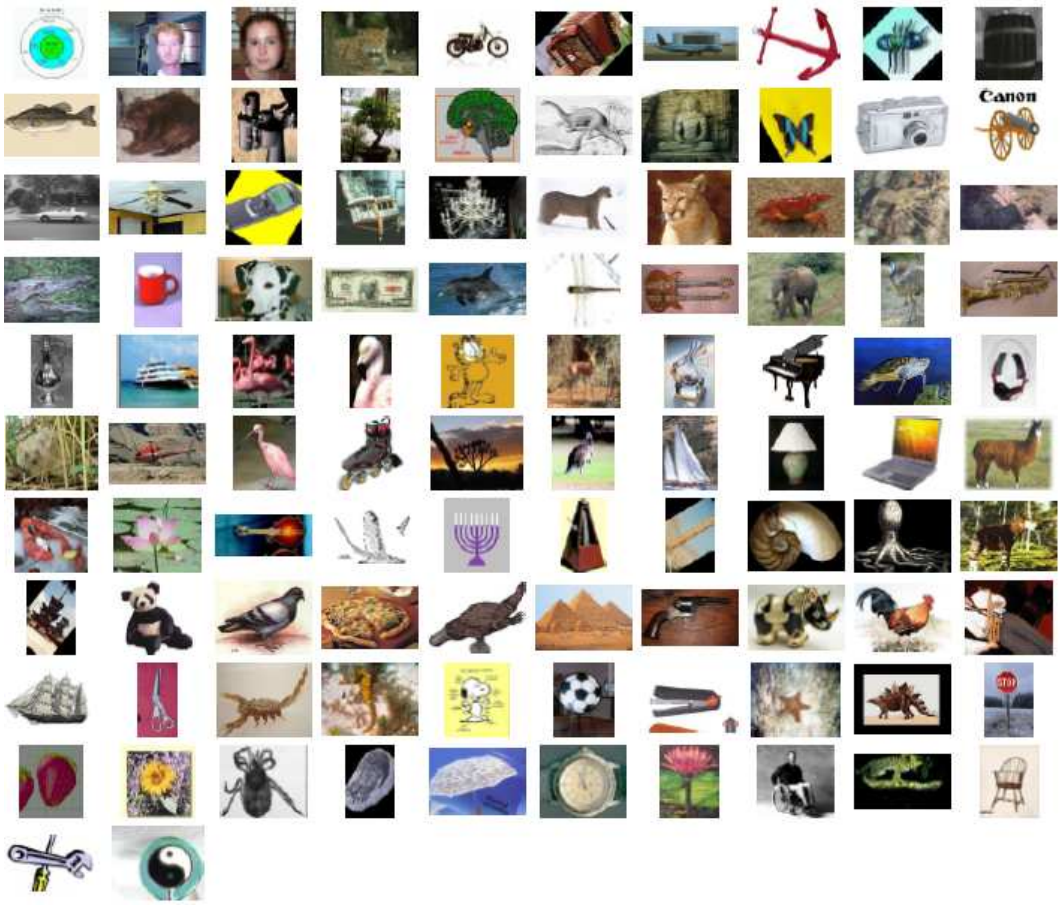


Figura 17. Clases Caltech

el diseño de la aplicación se ha necesitado de las herramientas Visual C++, para la implementación de la interfície, OpenCV y IPL, para el tratamiento de imágenes y la librería DirectShow de DirectX, utilizado para la visualización de las pruebas por su es potente variedad de formatos soportados.

Para la comparativa de resultados del sistema implementado en este proyecto usaremos otros métodos para comparar el descriptor BSM: ART [8], Zoning [9], Zernique[10], y descriptores de curvatura CSS [11] para el estándar MPEG [19][20][21]. Los parámetros de los descriptores corresponden a los proporcionados por los autores.

Para probar el rendimiento de los descriptores BSM, la comparativa se aplica sobre un conjunto de 23 clases MPEG07 y otro conjunto de 101 clases Caltech. Para demostrar la eficacia de los métodos de ECOC, la clasificación se realiza por medio de 3 vecinos más cercanos [18], lo que además nos permite comparar los descriptores con robustez independientemente del sistema de clasificación usado. De esta manera veremos la utilidad de la categorización mediante Adaboost y ECOC para problemas multi-clase.

7.3.1. Clasificación MPEG07

Los detalles de los descriptores utilizados para las comparaciones con objetos MPEG07 son los siguientes:

- Descriptor BSM de longitud 16×16 regiones. El tamaño óptimo de la resolución borrosa es de 16×16. Está resolución se calcula aplicando la validación cruzada sobre el conjunto de entrenamiento, utilizando un 10 % de las muestras para validar los diferentes tamaños de $n \in (8, 12, 16, 20, 24, 28, 32)$ (ver Fig.18). El tamaño seleccionado es el conjunto que alcanza el más alto rendimiento en el entrenamiento, definida la resolución borrosa óptima por el grado de confusión sobre la distorsión de los objetos de la base de datos. Los valores obtenidos mediante validación cruzada se observan en la Fig. 19.

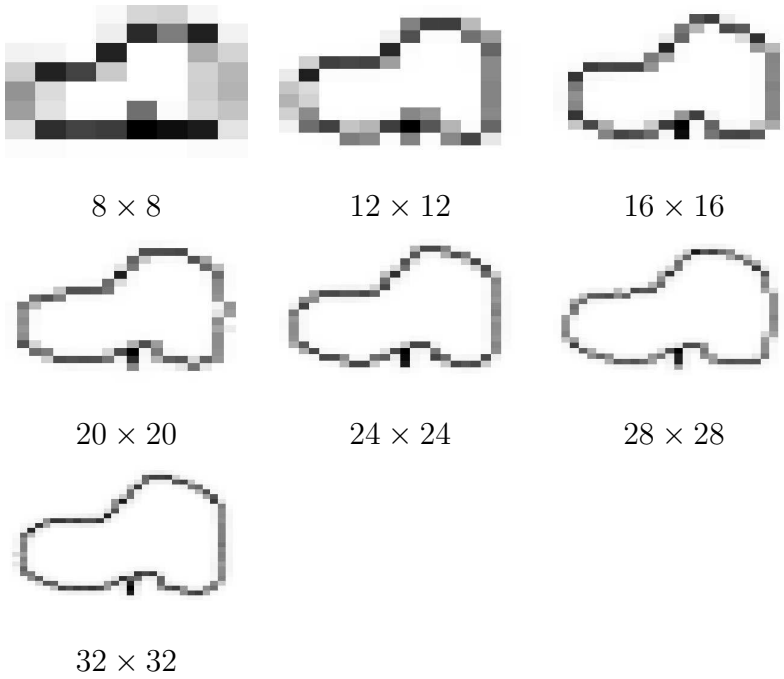


Figura 18. Resolución BSM para una muestra MPEG07

- Los parámetros para el descriptor ART son de orden radial con valor 2 y orden angular con valor 11.

- En cuanto a Zernique, 7 momentos se utilizan para calcular el descriptor.

- Para el descriptor de curvatura CSS se usará un longitud de 200 con un valor inicial de 1 σ, aumentando en uno el espacio del descriptor.

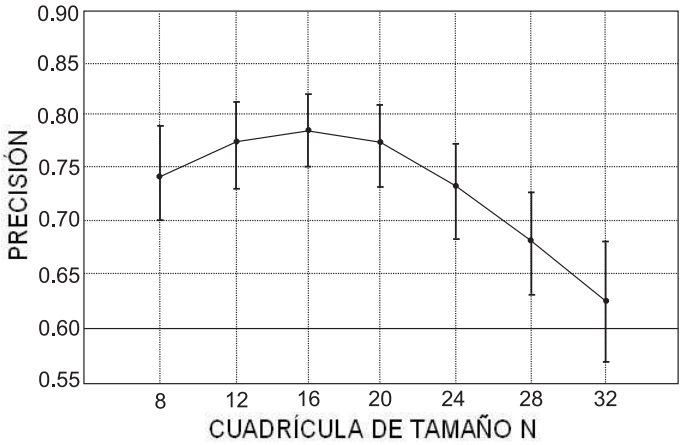


Figura 19. Entrenamiento de clases MPEG07 según resolución BSM

Para este experimento, empezamos la clasificación usando las 3 primeras clases de la Fig. 16. Reiterativamente, en cada paso se añade una clase, y la clasificación se repite hasta procesadas las 23 categorías. El objetivo principal es analizar el desempeño de las técnicas cuando el número de clases aumenta. Los resultados de el experimento se muestran en el Gráfico 20. Viendo el gráfico, se observa que el descriptor BSM alcanza el mejor rendimiento para cualquier número de clases en el sistema de clasificación. Además, un punto importante, es que su rendimiento no disminuye al tiempo que aumenta significativamente el número de clases, obteniendo resultados alrededor del 80 % en todos los casos. El segundo descriptor, Zernique, ofrece un rendimiento similar al BSM cuando el número de clases es pequeño, pero disminuye sustancialmente con el aumento del número de categorías. Por último, los descriptores Zoning, CSS, y ART ofrecen los peores resultados de clasificación en este problema. Esto puede ser debido a que estos descriptores funcionan solo en un ámbito local, y la base de datos está llena de variaciones de forma. Este hecho también afecta al descriptor CSS, ya que los puntos de curvatura varían debido a las grandes variaciones de forma entre los objetos.

Con el fin de validar los descriptores de forma independientemente del sistema, la clasificación de los 23 clase MPEG07 se realiza utilizando una estrategia simple de 3 vecinos más cercanos basada en la distancia Euclidana. Los resultados se muestran en la Fig. 21. Se puede observar una considerable disminución del

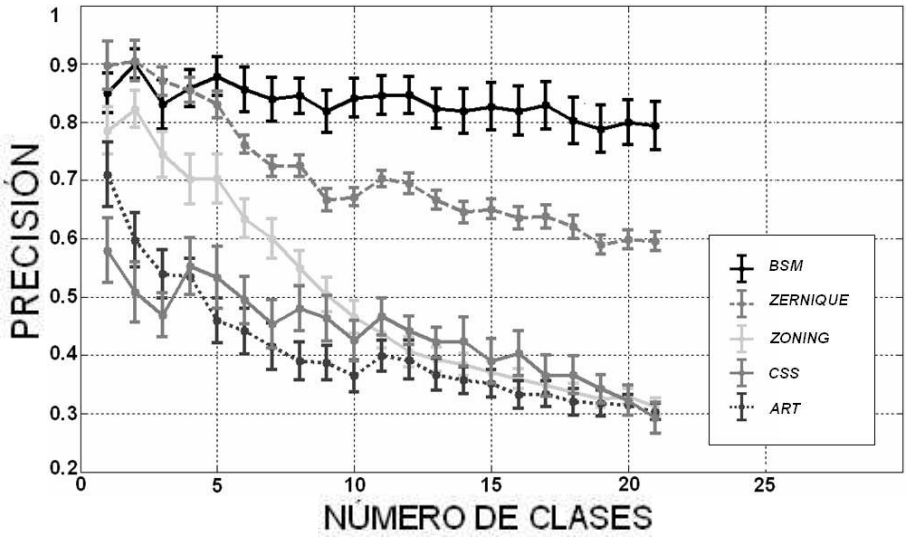


Figura 20. Clasificación de precisión por los sistemas de los objetos de la librería MPEG07

rendimiento para todos los descriptores. Esto es debido al hecho de que todas las características contribuyen a la decisión final, incluyendo errores en distancias que produzca pérdidas en muchas muestras clasificadas. Sin embargo, cabe observar que la reducción en el caso del descriptor BSM es menos considerable, y alcanza mejores rendimientos.

Descriptor	3-NN
BSM	**72.96 ± 3.52**
Zernique	52.67 ± 5.14
Zoning	21.74 ± 2.91
CSS	29.50 ± 5.65
ART	21.30 ± 3.53

Figura 21. Clasificación de 23 categorías MPEG07 usando 3 vecinos más cercanos

7.3.2. Clasificación Caltech 101

Los detalles de los descriptores utilizados para las comparaciones con objetos Caltech 101 son los siguientes:

- Para el descriptor BSM de longitud 16×16 regiones. También utilizando un 10 % de las muestras para validar los diferentes tamaños de $n \in (8, 12, 16, 20, 24, 28, 32)$. Las muestras a diferentes tamaños se muestran el la Fig. 22. En este caso el valor óptimo es de 16×16, tal es como se observa en la Fig. 24.

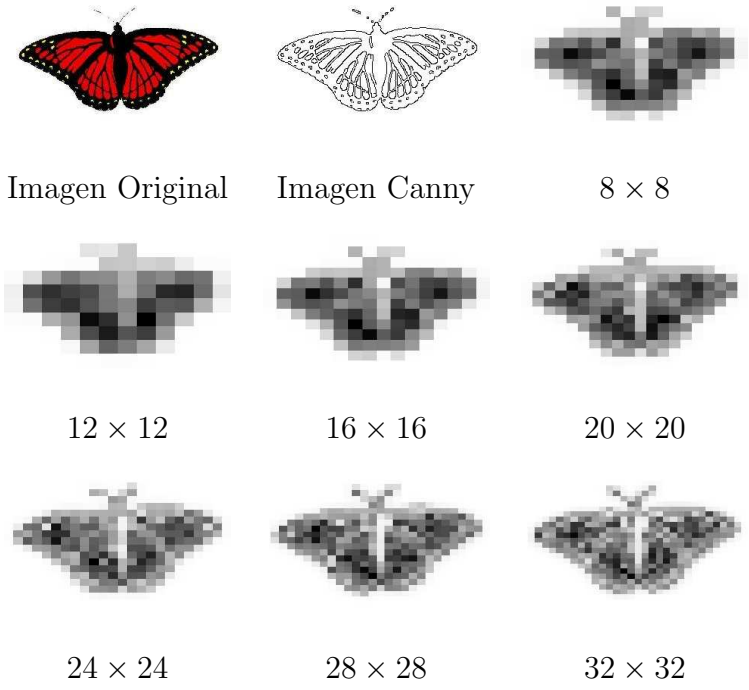


Figura 22. Resolución BSM para una muestra Caltech 101

- Para los otros descriptores para la comparativa seguiremos cogiendo los mismos valores que en el caso de la clasificación MPEG07 del apartado anterior.

Descriptor	3-NN
BSM	**62.25 ± 3.21**
Zernique	56.36 ± 5.02
Zoning	49.72 ± 2.35
CSS	23.69 ± 5.12
ART	30.45 ± 3.10

Figura 23. Clasificación de 101 categorías Caltech usando 3-Nearest Neighbor

Con el fin de validar los descriptores de forma independientemente del sistema, la clasificación la realizamos sobre 15 categorías de la Caltech, utilizando la misma estrategia vista en el experimento anterior. Las muestras de las 15 clases se muestran en la Fig. 25 y los resultados se muestran en la Fig. 23. Se puede observar que el descriptor BSM es claramente superior ante los otros descriptores en la clase de la clasificación de la categoría de la base de datos pública.

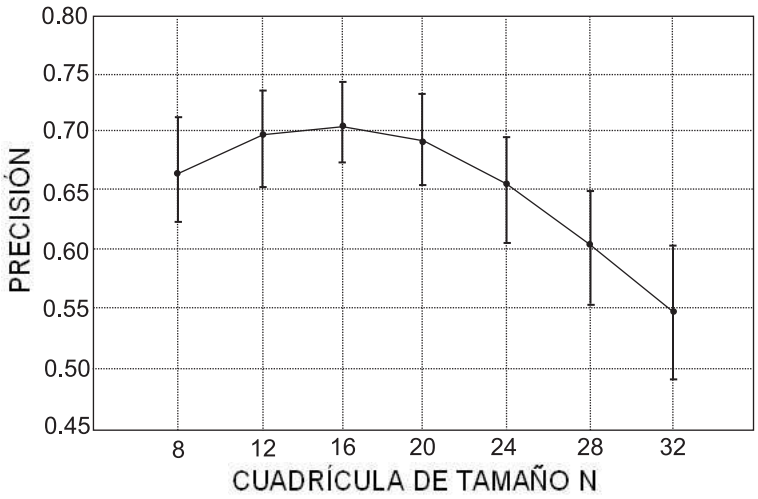


Figura 24. Entrenamiento de clases Caltech 101 según resolución BSM

Figura 25. 15 clases Caltech

7.3.3. ORMGPlayer

En este último apartado de los experimentos se muestra la extracción de los Descriptores BSM mediante el uso de la interfície implementada ORMGPlayer.

Ejemplos, para algunas muestras de la base de datos de descriptores BSM MPEG y Caltech se muestran en la Fig. 26 y Fig. 27, respectivamente.

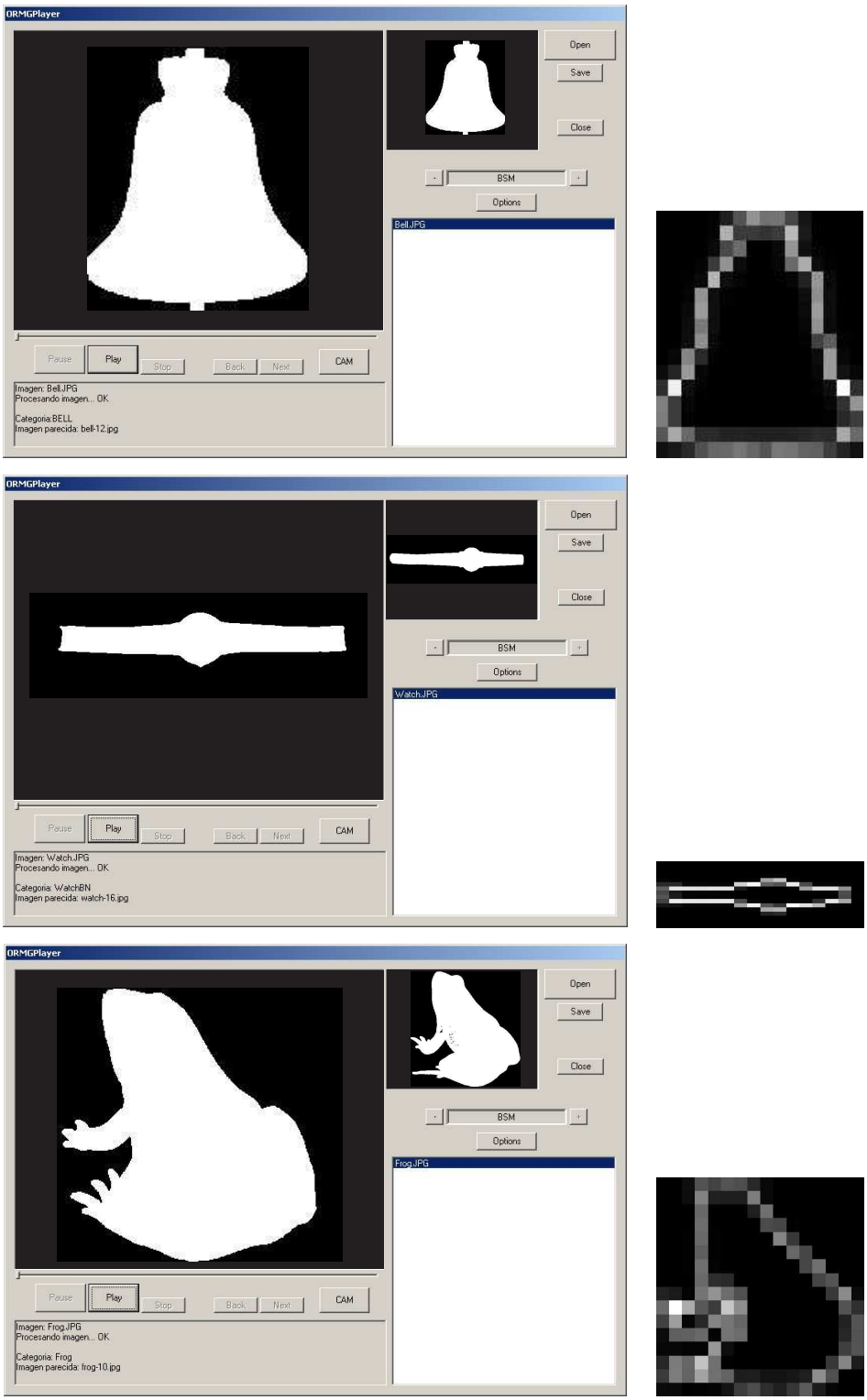


Figura 26. Aplicación ORMGPlayer con imágenes MPEG07

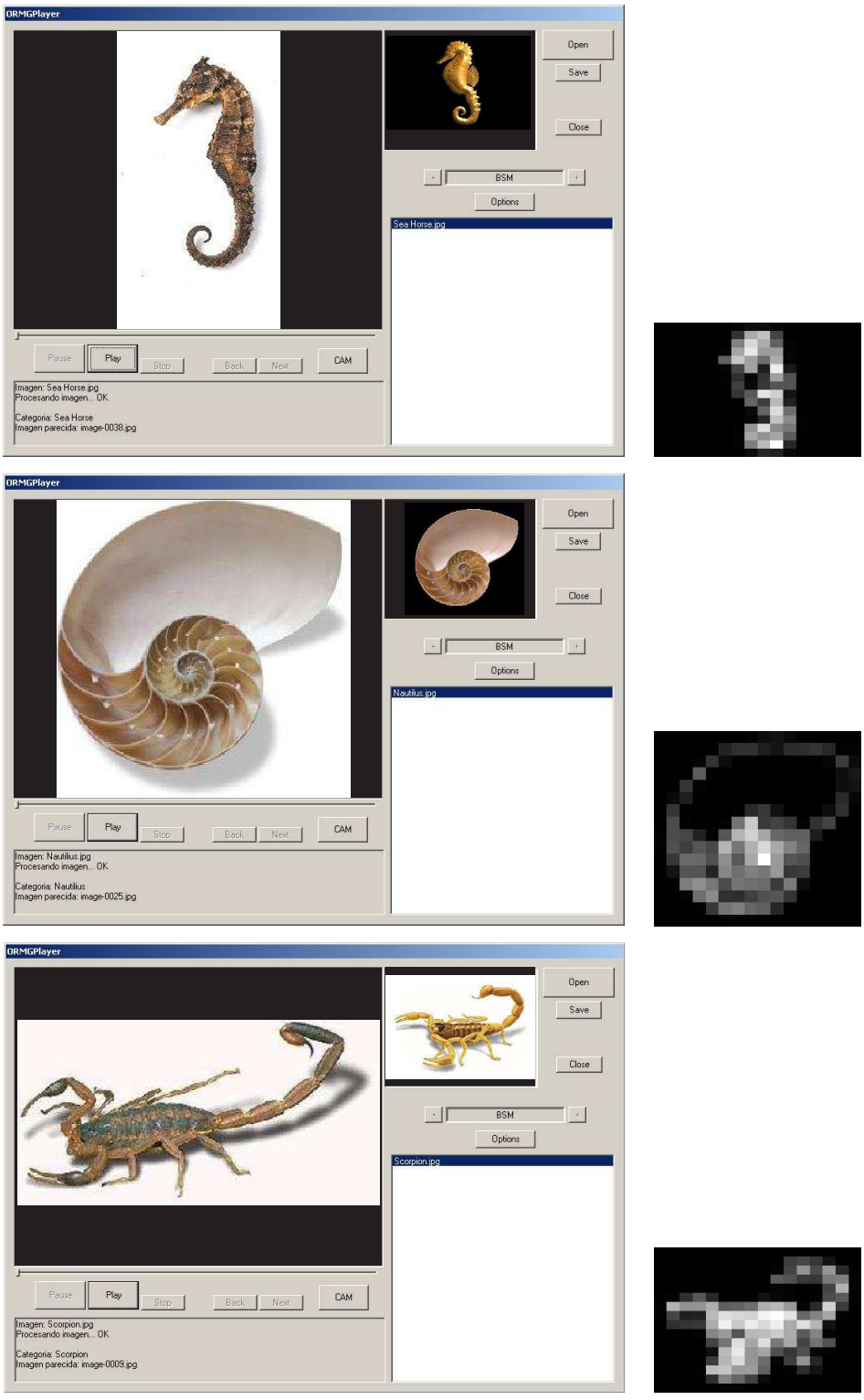


Figura 27. Aplicación ORMGPlayer con imágenes Caltech

8. Planificación

En la tabla siguiente veremos la planificación temporal del desarrollo de nuestro proyecto, remarcando que se ha extendido a lo largo de un año universitario. Podemos ver como se han ido consiguiendo, poco a poco, todas las expectativas iniciales.

LECTURA BIBLIOGRÁFICA	20h
ESTUDIO DE VIABILIDAD	20h
IMPLEMENTACIÓN DESCRIPTOR BSM	
Creación proyecto en blanco	2h
Búsqueda de información y material necesario (imágenes)	15h
Implementación BSM	45h
Depuración de código	15h
Pruebas	20h
IMPLEMENTACIÓN CLASIFICADOR ADABOOST	
Implementación Adaboost	10h
Implementación diseño ECOC	
Codificación ECOC 1vs1	10h
Decodificación Euclidiana	15h
Integración Adaboost + ECOC	5h
Pruebas	15h
IMPLEMENTACIÓN ORMGPlayer	
Creación proyecto en blanco	3h
Implementación Aplicación MFC	30h
Implementación Aplicación DirectX	25h
Depuración de código	20h
Pruebas	10h
INTEGRACIÓN DE BSM + ADABOOST + ECOC EN ORMGPLAYER	
Pruebas	15h
EXPERIMENTOS DEL SISTEMA DE RCONOCIMIENTO	
BSM + Adaboost + ECOC	20h
REDACCION DE LA MEMORIA	60h
TOTAL	**385h**

Figura 28. Planificación Temporal

9. Conclusiones

En este proyecto se presenta una estrategia para reconocer múltiples objetos en imágenes digitales mediante características basadas en forma. En cuanto a la idoneidad del sistema presentado para hacer frente a una categorización multi-clase de objetos, varias prestaciones deben ser mencionadas: el descriptor BSM presentado define una función de densidad de probabilidades de la forma de un objeto. El contorno del objeto es parametrizado con un conjunto de probabilidades que codifican su variabilidad espacial, siendo robusto contra deformaciones rígidas y elásticas. Se propone Adaboost para el aprendizaje de características con mayores zonas discriminativas para mejorar la división entre categorías de objetos. También se han usado los Códigos Correctores de Errores, cuyo marco tiene como propiedad corregir posibles errores producidos en el proceso de clasificación de los clasificadores binarios, a la vez que permite al sistema hacer frente a una categorización de problemas multi-clase. La evaluación del sistema se lleva a cabo con 23 clases de la base de datos pública MPEG07 (objetos binarios) y 101 clases de la también base de datos pública Caltech (imágenes en escala de grises y color), mostrando un gran rendimiento en los casos de alta relación visual entre las clases y variabilidad entre objetos de una misma clase. La ejecución del sistema completo, una vez entrenado, permite además ser usado en aplicaciones que funcionen en tiempo real. El sistema BSM está implementado en una clase C++ para que pueda ser utilizada por otros programadores y la aplicación ORMGPlayer se deja con código abierto para poder añadir futuros métodos implementados.

A. Contenido del CD

- Ejecutables: Contiene los archivos ejecutables del sistema como los instaladores de OpenCV, IPL y las librerías y cabeceras de las DXSDK.
- Código: Contiene la clase CBSM implementada.
- Datos: Muestras de imágenes de las clases MPEG07 y Caltech 101.
- Memoria: Contiene el archivo PDF con la memoria del proyecto.

B. Diseño del proyecto usando Visual C++, con librerías MFC, OpenCV, IPL y DirectShow

Para la creación de un proyecto en blanco para una programación mediante librerías OpenCV, IPL y DirectX se seguirán los siguientes pasos: se tendrá que instalar los ejecutables de OpenCV e IPL en las carpetas:

$C:\backslash Archivos\ de\ Programa\backslash Intel\backslash$ para la instalación de las IPLImage y,

$C:\backslash Archivos\ de\ Programa\backslash OpenCV\backslash$ para la instalación de las OpenCV.

Primero de todo, insertaremos las librerías necesarias en el proyecto VC++, para ello, seleccionaremos en el menú 'Project', la opción 'Settings' i clicando en la pestaña 'Link tab' añadimos en el campo 'Object/library modules' lo siguiente: ipl.lib cv.lib cxcore.lib highgui.lib cvaux.lib (ver Fig. B.1)

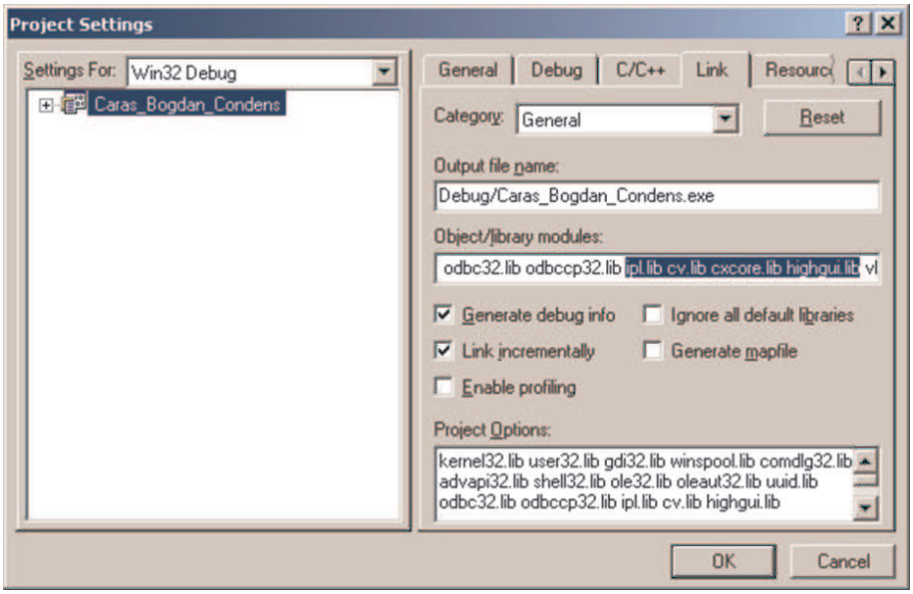


Figura B.1. Configuración VC++

Para incluir las cabeceras, desde el menú 'Tools' elegimos 'Options' y clicaremos sobre la pestaña 'Directories'. En el campo 'Show directories for' seleccionar 'Include files', e insertar los siguientes directorios (ver Fig. B.2):

$C:\backslash Archivos\ de\ Programa\backslash Intel\backslash Plsuite\backslash include$

$C:\backslash Archivos\ de\ Programa\backslash Intel\backslash OpenCV\backslash cv\backslash include$

$C:\backslash Archivos\ de\ Programa\backslash Intel\backslash OpenCV\backslash otherlibs\backslash highgui$

$C:\backslash Archivos\ de\ Programa\backslash Intel\backslash OpenCV\backslash cxcore\backslash include$

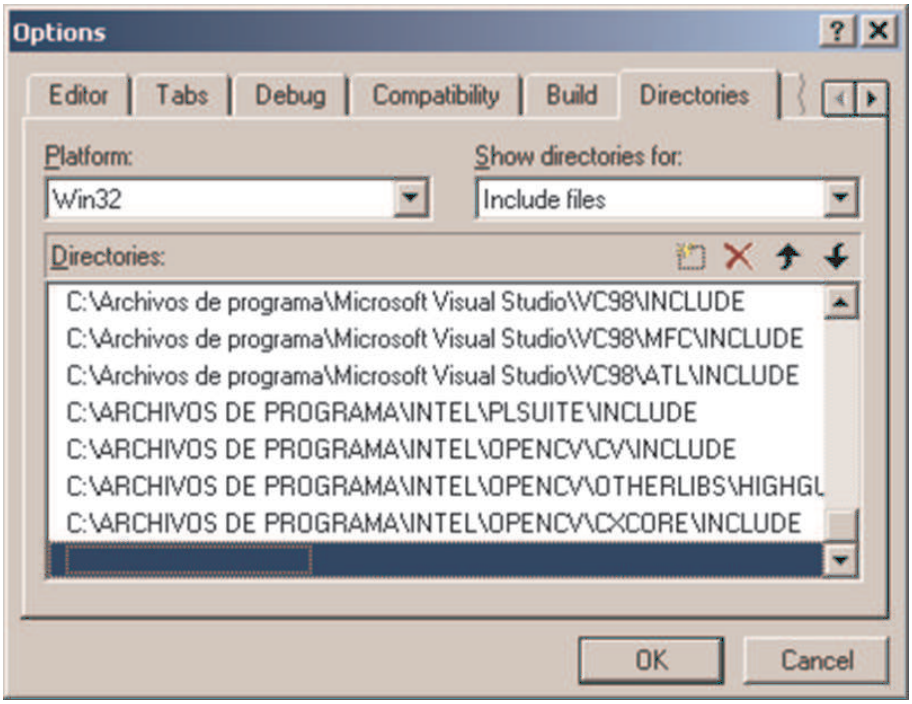


Figura B.2. Configuración Includes en VC++

Para incluir las librerías, desde el menú 'Tools' elegir 'Options' y clicar sobre la pestaña 'Directories'. En el campo 'Show directories for' seleccionar 'Library files', e insertar los siguientes directorios (ver Fig. B.3):

$C:\backslash Archivos\ de\ Programa\backslash Intel\backslash Plsuite\backslash lib\backslash MSVC$

$C:\backslash Archivos\ de\ Programa\backslash OpenCV\backslash lib$

En el último paso de la configuración consiste en añadir los directorios que contienen las DLL's en 'Environment Variables', ubicado en el Panel de Control, seleccionando 'System', 'Advanced' y finalmente 'Environment Variable'. En el cuadro diálogo editar los siguientes rutas (ver Fig. B.4):

$C:\backslash Archivos\ de\ programa\backslash Intel\backslash OpenCV\backslash bin$

$C:\backslash Archivos\ de\ programa\backslash Intel\backslash plsuite\backslash bin$

Finalizados todos los pasos se tendrá la aplicación, Visual C++, para una programación mediante OpenCV, IPL y DirectX.

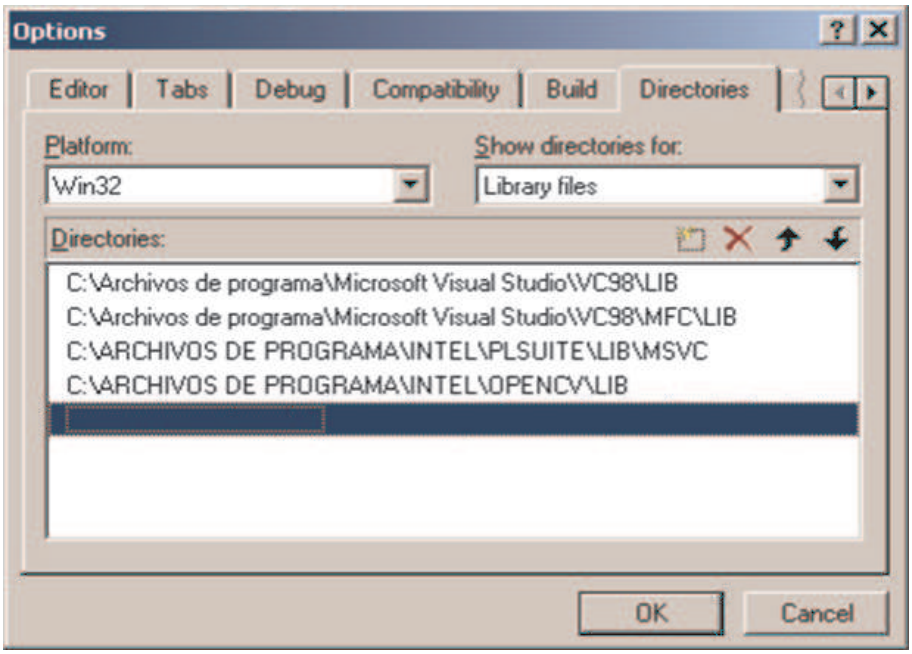


Figura B.3. Configuración de librerías en VC++

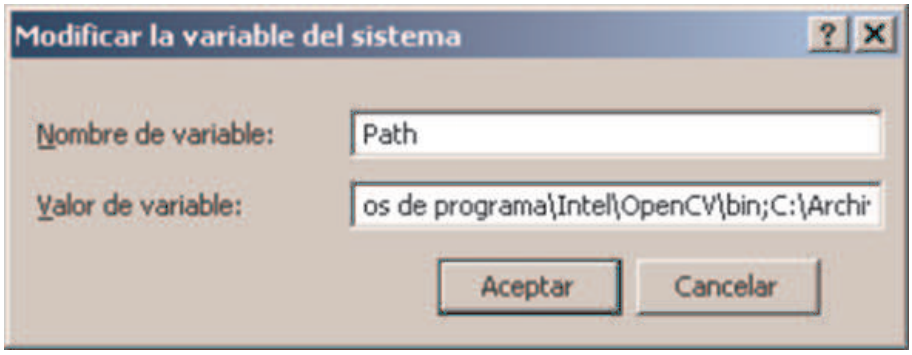


Figura B.4. Configuración de directorios en VC++

Referencias

[1] Viola P. & Jones M.: Rapid Object Detection using a Boosted Cascade of Simple Features, Computer Vision and Pattern Recognition, 2001

[2] T. Kikuchi and S. Abe. Error correcting output codes vs. fuzzy support vector machines. In ANNPR, 2003.

[3] K. Crammer and Y. Singer. On the learnability and design of output codes for multi-class problems. In Machine Learning, volume 47, pages 201-233, 2002.

[4] T. Dietterich and G. Bakiri. Solving multiclass learning problems via error-correcting output codes. In Journal of Artificial Intelligence Research, volume 2, pages 263-286, 1995.

[5] www.itu.int/itunews/manager/display.asp?lang=es&year=2005&issue=09 &ipage=things&ext=html

[6] L. Devroye, L. Gyor, and G. Lugosi. A probabilistic theory of pattern recognition. In Berlin: Springer-Verlag, 1996.

[7] http://www.cis.temple.edu/ latecki/research.html

[8] liris.cnrs.fr/Documents/Liris-1814.pdf

[9] www.springerlink.com/index/6m404l1117050775.pdf

[10] www.springerlink.com/index/e3n3085646621822.pdf

[11] www.springer.com/computer/computer+imaging/book/978-1-4020-1233-4?detailsPage=toc

[12] http://www.vision.caltech.edu/html-files/archive.html.

[13] A. Torralba, K. Murphy, and W. Freeman. Sharing visual features for multiclass and multiview object detection. In CVPR, volume 2, pages 762-769, 2004.

[14] jmlr.csail.mit.edu/papers/volume5/rifkin04a/rifkin04a.pdf

[15] T.Hastie and R.Tibshirani. Classification by pairwise grouping. NIPS, 26:451-471, 1998.

[16] E. Allwein, R. Schapire, and Y. Singer. Reducing multiclass to binary: A unifying approach for margin classifiers. volume 1, pages 113-141, 2002.

[17] www.pages.drexel.edu/ weg22/cantut.html

[18] Prtools toolbox. www.prtools.org, 2007. Faculty of Applied Physics, Delft University of Technology, The Netherlands.

[19] D. Zhang and G. Lu. Review of shape representation and description techniques. Pattern Recognition, 37:1-19, 2004.

[20] W. Kim. A new region-based shape descriptor. Technical report, Hanyang University and Konan Technology, 1999.

[21] Standard MPEG. ISO/IEC 15938-5:2003(E).

[22] T. Windeatt and R. Ghaderi. Coding and decoding for multi-class learning problems. In Information Fusion, volume 4, pages 11-21, 2003.

[23] Martin Pelikan, David E. Goldberg, and Erick Cantu-Paz. Learning machines. In McGraw-Hill, 1965.

[24] E. Allwein, R. Schapire, and Y. Singer. Reducing multiclass to binary: A unifying approach for margin classifiers. volume 1, pages 113-141, 2002.