PROYECTO FIN DE CARRERA

Uso de ontologías para guiar el proceso de segmentación de imágenes

Autor:

Luis Miralles Pechuán

Directores:

Rodrigo Martínez Béjar

Jesualdo Tomás Fernandez Breís

FACULTAD DE INFORMÁTICA

UNIVERSIDAD DE MURCIA

Murcia, febrero de 2007

RESUMEN

El objetivo de este proyecto consiste en potenciar el procesamiento de imágenes mediante el uso de tecnología ontológica.

Naturalmente, para poder emplear bien las ontologías en este dominio, el primer paso consistirá en familiarizarse con el procesamiento de imágenes.
Dentro del tratamiento digital los temas en los que más hemos profundizado son: el funcionamiento de las imágenes, su representación y las distintas operaciones. En el Anexo se muestra una síntesis de lo aprendido.

Una vez disponemos de estos conocimientos podemos enfrentarnos al problema que posee dos partes.

En primer lugar, trataremos de facilitar la selección de imágenes extrayendo características automáticamente y apoyándonos en las ontologías.
A continuación, se pretenderá ayudar al usuario, empleando las ontologías, a seleccionar un método de segmentación adecuado para cualquier imagen que se quiera procesar.

Por lo tanto nos vamos a centrar en los métodos de segmentación y en los algoritmos de extracción de características de las imágenes.

De los métodos de segmentación de imágenes hemos seleccionado los más importantes y los hemos implementado mediante una librería de C++ llamada OpenCv. Se ha desarrollado una aplicación en el entorno Visual Studio para analizar los resultados e identificar qué algoritmo se comporta mejor en función del tipo de imagen y del objetivo de segmentación.

Respecto a la extracción de características, se han usado 10 dominios como referencia. En este apartado hemos aprendido a extraer algunas características que tienen las imágenes como el tono de los colores, la detección de líneas, la forma de los histogramas o la detección de patrones. Mediante este tipo de características hemos diferenciado los distintos dominios implementando algún algoritmo propio que nos permita diferenciar unas imágenes del resto. De esta forma, hemos anotado las características extraídas de cada imagen mediante el uso de una ontología con fines de reusabilidad, compartibilidad y extensibilidad.

Adicionalmente, hemos creado una ontología con los métodos de segmentación y otra con los algoritmos de extracción de características para la clasificación de imágenes.

Estas ontologías nos han servido para desarrollar una aplicación que intenta recoger la teoría que se expone a lo largo del proyecto.

ÍNDICE

Capítulo 1 INTRODUCCIÓN

Capítulo 2 ANÁLISIS DE OBJETIVOS Y METODOLOGÍA

Capítulo 3 DISEÑO Y RESOLUCIÓN DEL TRABAJO

CAPÍTULO 1

INTRODUCCIÓN

1.1.- Introducción

El proyecto se ha dividido en distintas secciones. Por orden serian la “Introducción”, “Análisis de objetivos y metodología”, “Diseño y resolución”, “Conclusiones”, “Referencias” y “Anexo”.

Respecto a la introducción hemos intentado explicar la importancia de las imágenes en la actualidad y haremos un resumen sobre los objetivos del proyecto, las partes que va a tener.

En el análisis de objetivos se muestra los objetivos que intentamos abarcar en un principio y la metodología que hemos empleado. También las herramientas que se han utilizado, sus principales ventajas e inconvenientes.

En el apartado de Diseño y resolución, hablamos de la extracción de características, la segmentación y las ontologías. En la extracción de características explicaremos los métodos que hemos implementado y las principales estrategias en la que nos hemos basado para poder clasificar una imagen intentando averiguar su dominio. Cuando hablamos de segmentación hemos aplicado algunos de los métodos más importantes, diciendo sus fundamentos y analizando los resultados que hemos obtenido. A cerca de las ontologías diremos una definición y mostraremos la ontología que hemos creado y algunos ejemplos para ilustrar.

Posteriormente hablaremos de las conclusiones y de las posibles ampliaciones que se podrían plantear en un futuro.

Enumeramos en las Referencias los principales documentos, libros, apuntes, páginas web, etc.

El Anexo recogen los fundamentos de las imágenes. Donde hablaremos de como esta formado el ojo humano, de su funcionamiento y de cómo se forman las imágenes. Daremos una introducción de la colimetría, que es la ciencia del color.

Para finalizar este capítulo contaremos como se adquieren las imágenes, cuales son los tipos de imágenes y los principales operadores.

1.2.- Imágenes

1.2.1- Importancia de las imágenes

El ser humano se sirve de los sentidos para iterar con el mundo en el que se encuentra. Los sentidos le permiten conocer la realidad. Mediante los sentidos las personas captamos información del mundo que nos rodea [Jomupe]. Podemos palpar objetos, identificar olores, escuchar sonidos, detectar los sabores y lo más importante podemos ver el espacio en el que nos encontramos.

De todos los sentidos el que más se desarrolla es la vista. Es el medio por el cual más información recibimos. Nos permite percibir y comprender el mundo que nos rodea y representa casi el setenta por ciento de la información que recibimos. Entre este tipo de información se encuentran la identificación de caras, la lectura, las imágenes, etc.

Las escenas que percibimos suelen ser tridimensionales (3D) y cuando las capturamos mediante dispositivos (cámaras fotográficas o de vídeo, pantallas de rayos X, etc.) obtenemos imágenes bidimensionales (2D). El ser humano interactúa con un mundo tridimensional, cuando queremos captar una parte mediante cualquier dispositivo normalmente obtenemos una imagen bidimensional.

Por todas estas razones las imágenes tienen cada vez más protagonismo en nuestra sociedad. Las fotografías personales, las videoconferencias, los mapas reales, el cine, las noticias por audio. Todos estos elementos tienen algo en común y es que almacenan imágenes. Por lo tanto tenemos mucho interés en investigar y desarrollar buenos sistemas para el tratamiento de imágenes.

1.2.2- Extracción de características

Cuando queremos clasificar imágenes debemos analizarlas previamente [Giga]. La información extraída de una imagen puede ser cualitativa que es la que intenta averiguar que objeto esta representado en la imagen, pero también puede ser cuantitativa, este tipo analiza distintas cosas en las imágenes tales como las posiciones, los colores, la intensidad, la textura, etc.

Cuando analizamos una imagen podemos tener distintos objetivos:

- Detección de objetos: Dada una imagen seleccionar en ella un objeto mediante distintas técnicas
- Seguimiento y correspondencia: Trata de encontrar la equivalencia de puntos entre dos imágenes (por ejemplo, imágenes en una secuencia de vídeo o en un par estéreo).
- Reconstrucción 3D: Se refiere a la extracción de información 3D de la escena, posiciones, ángulos, velocidades, etc.
- Segmentación: Vamos a profundizar en este punto en el apartado siguiente. Le dedicamos un apartado porque es uno de los temas centrales del proyecto.

1.2.3- Segmentación de imágenes

La segmentación de imágenes es la descomposición de una imagen en las partes que la componen, es decir, distinguir los objetos y el fondo de la imagen.

Se puede decir que en las imágenes están formadas por distintas zonas que son parecidas en algún aspecto (color, textura, intensidad, etc.). Con mucha frecuencia cada zona representa un objeto distinto. La segmentación intenta reconocer las distintas zonas en función de la conectividad, el gradiente, la dirección de los bordes, los momentos, etc.

La segmentación es completa cuando cada zona distinta en la imagen se identifica con un objeto y cuando no se cumpla será segmentación parcial.
Si queremos conseguir un buen resultado en la segmentación completa necesitaremos tener algún conocimiento sobre la imagen que estamos analizando. Podemos poner por caso que la imagen represente un abecedario, o se corresponda con unos signos conocidos, etc. El resultado suele ser una imagen binaria ya que nos limitamos a señalar si el píxel valdrá pertenece o no, al objeto. Aún no hay ningún consenso sobre la definición de imágenes y sus propiedades aunque si podemos encontrar algunos algoritmos. [DiIm]

Los algoritmos tratados se basan en tres propiedades:

a) Discontinuidad: Esto nos permite detectar en la imagen puntos sueltos, líneas y bordes.
b) Similitud: Podemos separar regiones, unir píxeles en el crecimiento por semillas y separar por la intensidad mediante la umbralización.
c) Conectividad: Una región es conexa si para cualquier par de píxeles de la región existe un camino de píxeles de esa misma región que los conecta. Un camino es una secuencia de píxeles adyacentes dentro de la imagen.

1.2.4- Referencias históricas

La primera aplicación que se conoce de las imágenes digitales fue en la industria del periódico, cuando las imágenes eran enviadas por un cable entre Londres y Nueva Cork. La introducción de la transmisión de las imágenes a través del cable fue a principios de 1920. En este periodo se redujo el tiempo de envío de imágenes de más de una semana a menos de tres horas. Un especializado equipo de impresión codificó las imágenes por cable y al final del trayecto las reconstruía. Algunos de los problemas de impresión se debían a la selección de las intensidades de grises para las imágenes. El método de impresión usado para obtener alcanza el año 1921 a favor de una técnica

basada en la reproducción fotográfica hecha de cintas perforadas en el telégrafo recibiendo terminal. Las mejoras en esos pocos años eran patentes. El sistema Bartlane llego a codificar hasta quince niveles de gris en 1929. Estos descubrimientos tampoco es que estén completamente relacionados con el mundo de la imagen digital. La imagen digital se ha desarrollado alrededor de la computadora.

Los primeros trabajos relacionados con la visión artificial datan de la década de 1950. El entusiasmo inicial fue tan grande debido principalmente a una gran confianza en las posibilidades de los ordenadores y porque para el hombre la acción de ver es una tarea sencilla, e igual debería suceder para los ordenadores. Años después, ese entusiasmo desapareció debido a los limitados avances y a las pocas aplicaciones existentes.

Aunque en los años sesenta se desarrollaron algoritmos que son utilizados hoy en día, como los detectores de bordes de Roberts (1965), Sobel (1970) y Prewitt (1970), su funcionamiento estaba muy limitado a un reducido número de imágenes y casos. Es por ello que en los años setenta hubo un abandono progresivo en la investigación.

A partir de la década de los ochenta se empieza a hacer hincapié en la extracción de características. Así se tiene la detección de texturas (Haralik ,1979), y la obtención de la forma a través de ellas (Witkin (1981)). En ese mismo año, 1981, se publican artículos sobre visión estéreo[1] (Mayhew y Frisby), detección del movimiento (Horn), interpretación de formas (Steven) y líneas (Hanade); o los detectores de esquinas de Kitchen y Rosendfeld (1982).

El trabajo más importante de esa década es el libro de David Marr (Vision: a computacional investigation into the human reprentation and processing of cisual information,(1982) donde se abordaba por primera vez una metodología completa del análisis de imágenes a través del ordenador.

Los principales motivos de este crecimiento se debe en gran parte a un enfoque más realista del problema a resolver (por ejemplo, empieza a llamarse

visión por computador en lugar de visión artificial), al desarrollo de los ordenadores (aumento de la capacidad de cálculo y disminución del precio) y hardware especializado en el procesamiento y tratamiento de imágenes.

1.3.- Ontologías

1.3.1- Introducción

El término ontología se originó en la filosofía. En ese contexto, hace referencia a un campo que se encarga de estudiar la naturaleza de la existencia, una rama de la metafísica interesada en identificar, en términos generales, los tipos de cosas que realmente existen y como describirlas. Por ejemplo, la observación de que el mundo está hecho de objetos específicos que pueden ser agrupados en clases abstractas basadas en propiedades compartidas es una afirmación ontológica típica.

En los años más recientes, el concepto de ontología ha sido raptado por la ciencia informática, principalmente por la rama de la gestión del conocimiento de la inteligencia artificial, dándole un significado distinto del original. En lugar de hablar del término general ontología, hablamos de “una ontología”.

Según la definición de T.R.Gruber, posteriormente refinada por R.Studer, *una ontología es una especificación formal y explícita de una conceptualización compartida*.

En esta definición, convertida ya en estándar, conceptualización se refiere a un modelo abstracto de algún fenómeno del mundo del que se identifican los conceptos que son relevantes; explícito hace referencia a la necesidad de especificar de forma consciente los distintos conceptos que conforman una ontología; formal indica que la especificación debe representarse por medio de

un lenguaje de representación formalizado; y compartida indica que la ontología describe un conocimiento aceptado por un grupo de usuarios.

Una definición más concreta de ontología la ofrece Weigand (1997): *una ontología es una base de datos que describe los conceptos en el mundo o en algún dominio, algunas de sus propiedades, y como los conceptos se relacionan entre sí*.

Por tanto, aunque en filosofía una ontología es una explicación sistemática de la existencia, en los sistemas basados en el conocimiento, lo que existe es exactamente lo que se puede representar, y lo que se representa, mediante un formalismo declarativo, se conoce, con el nombre de Universo de Discurso. El UoD de una ontología es el conjunto de objetos que están representados en ella y sobre los cuales se puede hablar y razonar.

En general, una ontología describe formalmente un dominio del discurso. Típicamente, una ontología consiste en una lista finita de términos y relaciones entre estos términos. Los términos denotan los *conceptos* importantes o *clases de objetos* del dominio. Por ejemplo, en una Universidad, podemos identificar los conceptos de personal, estudiantes, profesores, cursos, disciplinas, …

Las relaciones típicamente incluyen jerarquías de clases. Una jerarquía especifica que una clase C es subclase de otra clase C' si cada instancia de C está también incluida (o es instancia de) C'. Por ejemplo, todos los profesores son miembros del personal de la universidad, por lo que la clase *Profesor* será una subclase de la clase P*ersonal*.

Aparte de las relaciones *subclase de*, las ontologías pueden incluir información como:

- Propiedades. Por ejemplo, la propiedad *enseña*, que relaciona un profesor con un alumno, o la propiedad *titulación*, que relaciona un profesor con su titulación académica.

- Restricciones de valor. Por ejemplo, solamente los profesores pueden impartir cursos.
- Afirmaciones sobre elementos disjuntos. Por ejemplo, los profesores y el personal de servicios son disjuntos (sin elementos en común).
- Especificación de relaciones lógicas entre objetos. Podríamos decir, por ejemplo, que cada departamento debe incluir al menos, a diez profesores.

En el contexto de la web, las ontologías proporcionan una *comprensión compartida de un dominio*. Tal compartición es necesaria para evitar las diferencias de terminología. Evitaríamos así que una aplicación hable de código postal, y otra de código de área, cuando el significado subyacente es el mismo. Otro problema es que dos aplicaciones pueden utilizar el mismo término, pero con diferentes significados.

Tales diferencias pueden eliminarse relacionando los conceptos de diferentes ontologías, proceso conocido técnicamente como *mapping*. Se pueden así aunar todas las diferentes ontologías de un dominio en una única ontología.

Tradicionalmente se distingues tres tipos fundamentales de ontologías:

- *Ontologías de un dominio*, en las que se representa el conocimiento especializado pertinente de un dominio o subdominio, como la medicina, las aplicaciones militares, la cardiología o la oncología.
- *Ontologías genéricas*, en las que se representan conceptos generales y fundacionales del conocimiento como las estructuras parte/todo, la cuantificación, los procesos o los tipos de objetos.
- *Ontologías representacionales*, en las que se especifican las conceptualizaciones que subyacen a los formalismos de representación del conocimiento, por lo que también se denominan meta-ontologías.

En Inteligencia Artificial existe una larga tradición de desarrollar y utilizar lenguajes de ontologías. Ellos son la base sobre la que se construye la

investigación de la Web Semántica (de manera general, la web semántica persigue una representación del contenido de la web que sea más fácilmente procesable por una máquina y el uso de técnicas inteligentes para sacar ventaja de estas representaciones).

Por el momento, los lenguajes de ontologías más importantes con los que podemos contar son los siguientes:

- XML. Proporciona una sintaxis que permite la estructuración de los documentos, es decir, describe las reglas que se deben seguir para escribir un documento. Sin embargo, no impone restricciones semánticas sobre el significado de estos documentos.
- XML Schema es un lenguaje para restringir la estructura de los documentos XML. Define las reglas que deben seguirse en la creación de un documento XML (antes definidas mediante DTDs).
- RDF es un modelo de datos (como pueda serlo el modelo entidad-relación) para describir objetos (recursos) y las relaciones entre ellos, proporcionando una semántica simple. Aunque RDF no está basado en XML, si que tiene una sintaxis de tipo XML para la representación de los modelos.
- RDF Schema es un lenguaje de descripción para describir propiedades y clases de recursos RDF, con la semántica necesaria para construir mediante generalización jerarquías de tales propiedades y clases. RDF Schema está basado en RDF. Se puede ver como un lenguaje primitivo, básico, para escribir ontologías, pero que no permite describir relaciones complejas entre objetos.
- OWL es un lenguaje de descripción más rico que el anterior, para describir clases, propiedades, relaciones entre clases, cardinalidad, igualdad, características de propiedades, clases enumeradas, …

Las ontologías, expresadas en cualquiera de los lenguajes, nos permitirán realizar lo que se conoce como *anotación semántica* de documentos. La notación consiste en asociar un elemento ontológico, como pueda ser un

concepto, un atributo, una relación, a un fragmento de texto de un documento, o a todo el documento. De esta manera se da un significado al texto anotado dentro del dominio modelado por la ontología. El proceso de anotación pretende dar significado a los documentos, y conseguir que sea más fácilmente interpretable y manejable por los sistemas informáticos. Uno de los objetivos de la web semántica es precisamente lograr que la información contenida en la web sea más fácilmente procesable por una máquina. Pero los documentos que se pueden anotar no tienen por qué ser solamente HTML, sino también texto plano, XML, etc.

La anotación no modifica el documento original, tan solo añade a él un conjunto de marcas, al estilo de las etiquetas HTML o XML, indicando que ese texto contiene una anotación. Cuando se anota un texto, además de elegir un determinado elemento ontológico, se puede suministrar otro tipo de información y guardar con la anotación.

Como lenguaje de ontologías, nuestro proyecto usa el lenguaje más completo de los mencionados, OWL, una propuesta de estandarización de lenguaje ontológico llamado OWL, especificado por un grupo de trabajo Web Ontology del consorcio W3C (http://www.w3.org/2001/sw/WebOnt/), que ayudaría a solucionar los impedimentos actuales para la construcción cooperativa de ontologías entre diferentes plataformas de construcción ontológica que usen diferentes modelos expresados en OWL.

1.3.2- Referencias históricas

La idea de que el lenguaje natural puede llegar a ser tratado de forma automática nace de un programa de investigación, que se remonta a principios del siglo XX, cuyo objetivo consistía en reconstruir el razonamiento matemático utilizando la lógica, y cuyos resultados se ponen de manifiesto en los trabajos de *Frege*, *Russell*, *Wittgenstein*, *Tarski*, *Lambek* y *Carnal*, que nos permiten

contemplar el lenguaje como un sistema formal susceptible de ser procesado automáticamente.

No sería sin embargo hasta los años cuarenta, con la aparición de los primeros rudimentarios ordenadores, que se podrían poner en práctica estas ideas. Desde entonces hasta hoy en día, la investigación en este campo se ha ido polarizando entre las aproximaciones empíricas guiadas por datos, que se basan en métodos estadísticos y de ensayo/error, destinadas a producir sistemas funcionales inmediatamente, y las aproximaciones racionalistas guiadas por el conocimiento, que involucran una profunda investigación lingüística y cuyo objetivo es la producción a largo plazo de soluciones más perfectas.

Las primeras aplicaciones en este campo se dieron, por tanto, durante el período de 1940-1960, teniendo como interés fundamental la traducción automática. Los experimentos en este sector, basados en la substitución de palabra por palabra, obtuvieron resultados rudimentarios. Surgió por tanto la necesidad de resolver ambigüedades sintácticas y semánticas, y asimismo, la consideración de información contextual, que en estos momentos se abordaron utilizando técnicas estadísticas basadas en la teoría de la información de *Shannon* y en técnicas criptográficas desarrolladas durante la Segunda Guerra Mundial.

Tras una primera decepción por la obtención de unos resultados bastante pobres al utilizar técnicas estadísticas mediante unos ordenadores muy poco potentes, en los años sesenta los intereses se desplazan hacia la comprensión del lenguaje. La mayor parte del trabajo realizado en este período se centró en técnicas de análisis sintáctico basadas en las contribuciones de *Noam Chomsky*, que introdujo la idea de las *Gramáticas Generativas* (unas descripciones formales basadas en reglas de las estructuras sintácticas).

Hacia la década de 1970 la influencia de los trabajos en inteligencia artificial fue decisiva, y se centró el interés en la representación del significado. Como resultado se construyó el primer sistema de preguntas-respuestas basado en

lenguaje natural. De esta época es ***Eliza***, que reproducía las habilidades conversacionales de un psicólogo. Para ello recogía patrones de información de las respuestas del cliente y elaboraba preguntas que simulaban una entrevista.

En las décadas de 1970 y 1980, ya superados los primeros experimentos, se hacen intentos de construir programas más fiables. Aparecen numerosas gramáticas orientadas a un tratamiento computacional, y experimenta un notable crecimiento la tendencia hacia la programación lógica. En Europa surgen intereses en la elaboración de programas para la traducción automática. Se crea el proyecto de investigación ***Eurotra***, que tiene como finalidad la traducción multilingüe. En Japón aparecen equipos dedicados a la creación de productos de traducción para su distribución comercial.

Los últimos años se caracterizan por la reaparición de las técnicas estadísticas gracias a la gran potencia de los ordenadores actuales, y se desarrollan formalismos adecuados para el tratamiento de la información léxica. Se introducen nuevas técnicas de representación del conocimiento cercanas a la inteligencia artificial y las técnicas de procesamiento utilizadas por investigadores procedentes del área de la lingüística e informática son cada vez más próximas. Surgen así mismo intereses en la aplicación de estos avances en sistemas de recuperación de información con el objetivo de mejorar los resultados en consultas a texto completo.

CAPÍTULO 2

ANALISIS DE OBJETIVOS Y METODOLOGÍA

2.1.- Objetivos del proyecto

2.1.1.- Objetivos generales del proyecto

- Desarrollo de algoritmos propios de extracción de características y el consiguiente análisis de los resultados sobre las distintas imágenes.

- Implementación de los principales métodos de segmentación sobre imágenes pertenecientes a diez dominios.

- Desarrollo de las ontologías necesarias para diseñar un sistema capaz de aplicar un algoritmo de segmentación adecuado elegido por el usuario y ayuda a la selección de imágenes con algunas características.

- Desarrollo de una aplicación en la que se implementen técnicas de visión artificial y se apliquen en función de un sistema basado en ontologías.

2.1.2.- Objetivos secundarios del proyecto

Dentro de cada objetivo principal podemos encontrar algunos objetivos secundarios:

De la extracción de características:

- Estudio de las técnicas de extracción de características de imágenes.
- Implementación de los algoritmos en el entorno Visual Studio, mediante la librería OpenCV con el lenguaje de programación C++.
- Comparación de los los resultados obtenidos sobre los distintos dominio y las distintas imágenes.

De los algoritmos de segmentación:

- Estudio de las distintas técnicas de segmentación y de los principales métodos.
- Implementación de los algoritmos de segmentación utilizando las funciones que proporciona la librería Open Cv desarrollada por intel.
- Comparación de las imágenes segmentadas y selección de los algoritmos que mejor funcionan para nuestro dominio.

De las ontologías:

- Diseño de una ontología para los algoritmos de segmentación.
- Diseño de una ontología para los métodos de extracción de características de las imágenes.
- Extracción de atributos entendibles por el usuario para facilitarle la interacción con el sistema.

Del desarrollo de la aplicación:

- Unir los métodos de extracción de características y los algoritmos de segmentación.
- Desarrollo de algoritmos de selección de los métodos anteriores basados en ontologías.
- Creación de un interfaz para poder interactuar con el usuario.
- Lectura y escritura de ficheros creando ontologías en Owl.

2.1.3.- Contenido del proyecto

Lo que pretendemos en este proyecto es encontrarle alguna aplicación a las ontologías al mundo de la visión por computador.

La metodología que se ha seguido durante el proyecto ha sido documentarse sobre los fundamentos del tratamiento de imágenes y sobre la visión por computador. Dentro de este tema hemos profundizado especialmente en la segmentación de imágenes, en los métodos de clasificación y en los principales métodos de su aplicación. Una vez estudiados, los hemos implementado mediante el Open Cv de Visual C++ y nos hemos creado las aplicaciones para probar su funcionamiento.

La siguiente fase del proyecto consiste en diseñar una ontología de los dominios en función de algunos criterios y realizar las funciones de proximidad. Tras haber realizado todas estas tareas deberemos probar nuestros métodos para nuevos conceptos. Les aplicaremos unos métodos u otros en función de la distancia semántica en la ontología.

La ontología es la parte más importante del proyecto ya que esta es nuestra especialidad. Nos hemos ayudado de las ontologías para seleccionar un método de clasificación eficiente para nuestro dominio y también para encontrar un buen algoritmo de segmentación para ese dominio.

Al final desarrollamos una aplicación con los conocimientos que se muestran a lo largo del proyecto.

2.2.- Herramientas utilizadas

2.2.1. - C++

C++ es un lenguaje imperativo orientado a objetos derivado del C. En realidad un superconjunto de C, que nació para añadirle cualidades y características de las que carecía. El resultado es que como su ancestro, sigue muy ligado al hardware subyacente, manteniendo una considerable potencia para programación a bajo nivel, pero se la han añadido elementos que le permiten también un estilo de programación con alto nivel de abstracción.

A efectos prácticos, dado el esfuerzo de compatibilidad desplegado en su diseño, puede considerarse que C++ es una extensión del C clásico. La definición "oficial" del lenguaje nos dice que C++ es un lenguaje de propósito general basado en el C, al que se han añadido nuevos tipos de datos, clases, plantillas, mecanismo de excepciones, sistema de espacios de nombres, funciones inline, sobrecarga de operadores, referencias, operadores para manejo de memoria persistente, y algunas utilidades adicionales de librería.

Ha experimentado un extraordinario éxito desde su creación. De hecho, muchos sistemas operativos, compiladores e intérpretes han sido escritos en C++ (el propio Windows y Java). Una de las razones de su éxito es ser un lenguaje de propósito general que se adapta a múltiples situaciones.

2.2.2. - Visual Studio

Visual Studio es un conjunto completo de herramientas de desarrollo para la generación de aplicaciones Web ASP.NET, Servicios Web XML, aplicaciones de escritorio y aplicaciones móviles. Visual Basic, Visual C++, Visual C# y Visual J# utilizan el mismo entorno de desarrollo integrado (IDE), que les permite compartir herramientas y facilita la creación de soluciones en varios lenguajes. Asimismo, dichos lenguajes aprovechan las funciones de .NET Framework, que ofrece acceso a tecnologías clave para simplificar el desarrollo de aplicaciones Web ASP y Servicios Web XML.

Hemos seleccionado Visual Studio porque es una herramienta muy completa con un entorno bastante intuitivo. Este entorno facilita en gran medida el desarrollo de las aplicaciones. Además, posee un potente depurador de incluido en la propia herramienta que permite marcar en el programa “Break points” y muestra una tabla con el valor actualizado de las variables.

Visual Studio incluye un amplio manual digital de consulta, que esta indexado donde podemos hacer búsquedas. También interactúa con este entorno. Simplemente con pulsar una tecla sobre una función saltara la ayuda de esa función en concreto.

Visual Studio es fácil de usar y se han creado en Internet foros donde poder preguntar las dudas. El más representativo es del de Yahoo llamado OpenCv.

2.2.3. - OpenCV

2.2.3.1- Introducción

Open Cv viene de las siglas Open source Computer Vision library, es una librería abierta desarrollado por Intel. Esta librería proporciona un alto nivel funciones para el procesado de imágenes. Estas librerías permiten a los programadores para crear aplicaciones poderosas en el dominio de la visión digital. OpenCV ofrece muchos tipos de datos de alto-nivel como juegos, árboles, gráficos, matrices, etc. OpenCV es opensource para poder funcionar en muchas plataformas.

OpenCv permite, Operaciones básicas, Procesado de imágenes y análisis, Análisis estructural, Análisis de movimiento, Reconocimiento del modelo, Reconstrucción 3d y calibración de la cámara, Interfaz gráfica y adquisición, etc.

OpenCV implementa una gran variedad de herramientas para la interpretación de la imagen. Es compatible con Intel Image Processing Library (IPL) que implementa algunas operaciones en imágenes digitales. A pesar de primitivas como binarization, filtrado, estadísticas de la imagen, pirámides, OpenCV es principalmente una librería que implementa algoritmos para las técnicas de la calibración (Calibración de la Cámara), detección de rasgos, para rastrear (Flujo Óptico), análisis de la forma (Geometría, Contorno que Procesa), análisis del movimiento (Plantillas del Movimiento, Estimadores), reconstrucción 3D(Transformación de vistas), segmentación de objetos y reconocimiento (Histograma, etc.).

El rasgo esencial de la librería junto con funcionalidad y la calidad es su desempeño. Los algoritmos están basados en estructuras de datos muy flexibles, acoplados con estructuras IPL; más de la mitad de las funciones ha sido optimizada aprovechándose de la Arquitectura de Intel.

2.2.3.2- Inconvenientes de OpenCV

Uno de los pocos inconvenientes que se pueden encontrar en ella sea en el caso del seguimiento de objetos, en el cual, el principal inconveniente que es que no ofrece un producto completo, tan sólo algunas piezas que sirven como base para montar sobre ellas un producto final.

Otro de los inconvenientes que tiene es la necesidad de utilizar la librería IPL para tener acceso a funciones de bajo nivel.

Sin embargo, la presencia de funciones muy interesantes, y las posibilidades ya comentadas que ofrece la librería hacen que estos inconvenientes no sean realmente significantes.

2.2.4- OWL

OWL (Ontology Web Language) [OwWe] surge del W3C como la búsqueda de un lenguaje de especificación de ontologías que sirva como estándar para todos los investigadores de la Web Semántica.

Se pueden definir clases mediante restricciones a otras clases, o con operaciones boleanas sobre otras clases, hay nuevas relaciones entre clases como la inclusión, disyunción y la equivalencia, se pueden definir restricciones de cardinalidad en propiedades o dar propiedades sobre las relaciones (transitiva, simetría) así como permitir clases enumeradas [DepInf].

Se decidió separar en tres niveles:

• OWL Lite: la versión más simple para los programadores principiantes. Permite la jerarquía de clasificación y las restricciones simples.

• OWL DL: esta versión ya tiene todo el vocabulario OWL completo. Las limitaciones son que las clases no son instancias ni tipos y los tipos no son ni

instancias ni clases. No permite restricciones de cardinalidad en propiedades transitivas. Posee gran expresividad sin perder las propiedades de completitud y decidibilidad.

• OWL Full: esta versión también incluye todo el vocabulario de OWL pero en este caso no hay limitaciones para explotar todo su potencial. Sin garantías computacionales.

OWL Full se considera la más completa de todas y se supone una extensión de DL que a su vez es una extensión de Lite, por lo que toda ontología correcta en OWL Lite es una ontología correcta en OWL DL, y toda conclusión correcta en OWL Lite es una conclusión correcta en OWL DL (pero no a la inversa). De la misma manera esto también ocurre con OWL DL y OWL Full respectivamente.

Desde febrero de 2004 una recomendación de W3C, por lo que la mayoría de desarrolladores e investigadores de la Web Semántica van a centrar ya sus esfuerzos en desarrollar herramientas y sistemas orientados a este lenguaje.

2.3.- Técnicas de segmentación de imágenes

En esta sección mostraremos los fundamentos teóricos de los algoritmos de segmentación que hemos implementado en nuestra aplicación. No vamos a profundizar en las demostraciones y últimos detalles porque no es nuestro objetivo prioritario. Simplemente daremos algunas pinceladas para ilustrar cada algoritmo.

2.3.1. – Umbralización

La umbralización es uno de los algoritmos de segmentación más sencillos. La idea es que muchas veces se representan objetos sobre una superficie del

mismo color [FuVi]. Esta técnica consiste en filtrar los píxeles que forman la imagen de forma que si superan un umbral se pondrán a 0. En caso contrario el se pondrán a 1 o no se modificará.

Dada un píxel g en una imagen en la posición(x,y):

$$g(x,y) = \begin{cases} 1 \Leftrightarrow f(x,y) > T \\ 0 \Leftrightarrow f(x,y) \leq T \end{cases}$$

La máxima dificultad radica en la elección del umbral. Esto puede hacerse de varias formas:

2.3.1.1. – Media

Este es un método de cálculo de umbral muy sencillo. Consiste en calcular la media de todos los píxeles.

2.3.1.2.- Búsqueda del Máximo y Mínimo

Este algoritmo consiste en seguir estos tres pasos:

- Localizar el máximo del histograma.
- Localizar el segundo máximo, se refiere al local. Máximo producto del valor del histograma por distancia.
- Localizar el mínimo entre ellos.

2.3.1.3. - Algoritmo de Otsu

Entendemos una imagen como una función de dos dimensiones f(x,y) = G. Donde G estará en el rango [0..255]. Los píxeles con valor de gris se representan como fi y el número de píxeles como N. La probabilidad de ocurrencia será:

$$p_i = \frac{f_i}{N}$$

Si solo hay dos niveles, se llama binarización, en esta caso estará C1, con niveles de gris [1, ..., t]; y C2 [t+1, ..., 255].

En el caso de la umbralización en dos niveles de una imagen (a veces llamada binarización), los píxeles son divididos en dos clases: C1, con niveles de gris [1, ..., t]; y C2, conniveles de gris [t+1, ..., L]. Entonces, la distribución de probabilidad de los niveles de gris para las dos clases son:

$$C_1 : \quad \frac{p_1}{\omega_1(t)}, \ldots, \frac{p_t}{\omega_1(t)}$$
$$C_2 : \quad \frac{p_{t+1}}{\omega_2(t)}, \frac{p_{t+2}}{\omega_2(t)}, \ldots, \frac{p_L}{\omega_2(t)}$$

donde

$$\omega_1(t) = \sum_{i=1}^{t} p_i \qquad \omega_2(t) = \sum_{i=t+1}^{L} p_i$$

Y la media para la clase C1 y la clase C2 es

$$\mu_1 = \sum_{i=1}^{t} \frac{i.p_i}{\omega_1(t)} \qquad \mu_2 = \sum_{i=t+1}^{L} \frac{i.p_i}{\omega_2(t)}$$

Sea μT la intensidad media de toda la imagen. Es fácil demostrar que

$$\omega_1.\mu_1 + \omega_2.\mu_2 = \mu_T \qquad \omega_1 + \omega_2 = 1$$

Otsu definió la variancia entre clases de una imagen umbralizada como

$$\sigma_B^2 = \omega_1.(\mu_1 - \mu_T)^2 + \omega_2.(\mu_2 - \mu_T)^2$$

Para dos niveles, Otsu el umbral óptimo t* se elige de manera que σB
2 sea máxima; esto es:

$$t^* = \underset{t}{Max}\{\sigma_B^2(t)\} \qquad 1 \le t \le L$$

2.3.1.4.- Detector de piel

Para que un píxel pertenezca a la piel, nos vamos a basar en un criterio de selección que se ha aplicado en otros casos con éxito. Nuestro algoritmo simplemente recorre todos y cada uno de los píxeles de la matriz que representa a la imagen. Para cada punto detecta si el píxel tiene el color de la piel y en caso de que negativo se pondrá en negro [UtLib].

2.3.2.- Detección de discontinuidades

Sabemos que representamos una imagen como una matriz. En el caso de una imagen de grises cada píxel se representara mediante un elemento de la matriz. Podemos detectar los píxeles que tienen distinto valor de los que le rodean aplicando distintas máscaras:

W_1	W_2	W_3
W_4	W_5	W_6
W_7	W_8	W_9

$$R = w_1 z_1 + w_2 z_2 + \ldots + w_9 z_9 = \sum_{i=1}^{9} w_i z_i$$

(a) Máscara

La máscara de la figura (a) es una zona de la imagen donde calculamos el valor de los distintos píxeles. R será el resultado suma de ellos. Los valores de W dependen de la operación que queramos realizar.

2.3.2.1.- Detección de puntos

La detección de puntos de una imagen se efectúa aplicando la máscara a un píxel y a sus vecinos. Esto es porque suponemos que el nivel de gris es distinto a los que forman las imágenes.

-1	-1	-1
-1	8	-1
-1	-1	-1

(a) Máscara para la detección de puntos aislados.

Nos damos cuenta de que en el caso de que la región sea homogénea el valor de R será igual a cero. El valor de R será mayor cuanto más difiera el píxel central al resto de sus vecinos.

2.3.2.2.- Detección de líneas

-1	2	-1
-1	2	-1
-1	2	-1

(a) Vertical

2	-1	-1
-1	2	-1
-1	-1	2

(b) -45º

-1	-1	-1
2	2	2
-1	-1	-1

(c) Horizontal

-1	-1	2
-1	2	-1
2	-1	-1

(d) 45º

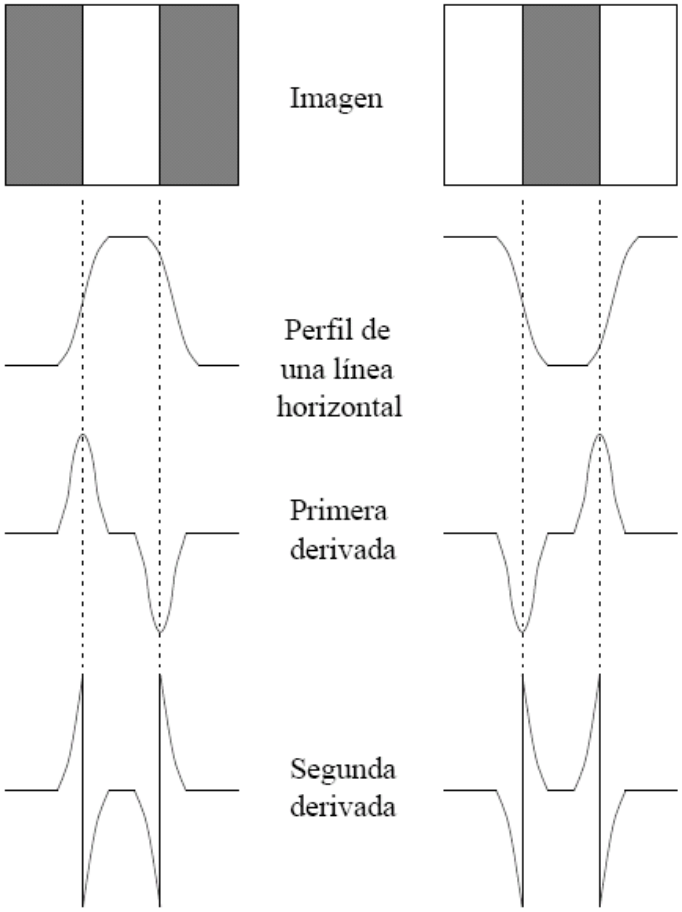
Si en cierto punto de la imagen $|R_i| > |R_j|$ para todo $i \neq j$, se dice que este punto estará asociado con la máscara de la dirección i. Si en un punto de la imagen, $|R_1| > |R_j|$, para j = 2, 3, 4 está mejor asociado con una línea horizontal.

Estas máscaras encuentran líneas de un píxel grueso. Si queremos encontrar líneas en una cierta dirección entonces aplicamos la máscara definida para esa

dirección. La suma de los coeficientes de la máscara es igual a cero indicando que la respuesta será cero en regiones homogéneas.

2.3.2.3.- Detección de bordes

Definimos borde como la frontera entre dos regiones con diferentes tonos o que tengan distintas regiones.Cuando queramos detectar bordes por las variaciones de los niveles de los tonos intentaremos encontrar cambios bruscos y quitar los valores constantes. Para encontrarlos nos apoyaremos en las derivadas [DiIm].



Detección de bordes por los operadores derivativos: (a) Una banda clara sobre un fondo oscuro, (b) Una banda oscura sobre un fondo claro.

El perfil se define perpendicular a la dirección del borde. La magnitud de la primera derivada se utiliza para detectar un borde y signo de la segunda derivada puede determinar si un píxel borde está en el lado claro u oscuro del borde.

La segunda derivada produce dos respuestas por cada borde. La línea que puede formarse uniendo el lado positivo y el negativo cruza por cero en el punto medio del borde.

Para calcular la primera derivada utilizamos la magnitud del operador gradiente en este punto y la segunda derivada se obtiene utilizando el laplaciano.

2.3.2.3.1.- Operadores gradiente

La primera derivada de una imagen digital está basada en varias aproximaciones del gradiente en 2D. El gradiente de una imagen f(x,y) la posición (x,y) esta definido como el vector:

$$\nabla F = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

En la práctica es común aproximar el gradiente con los valores absolutos:

$$\nabla f \approx | G_x | + | G_y |$$

La dirección del vector gradiente también es una cantidad importante. Si (x; y) representa el ángulo de dirección del vector (x; y), entonces del análisis de vectores.

$$\alpha(x, y) = tan^{-1}\left(\frac{G_y}{G_x}\right)$$

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

-1	0	1
-2	0	2
-1	0	1

-1	-2	-1
0	0	0
1	2	1

(a) Una región de una imagen de tamaño 3x3 píxeles. (b) Máscara usada para computar Gx en el punto central de la región (a). (c) Máscara para computar Gy en el mismo punto. Estas máscaras son referidas frecuentemente como operadores Sobel.

Gx = (Z3 + 2Z6 + Z9) - (Z1 + 2Z4 + Z7)
Gy = (Z7 + 2Z8 + Z9) - (Z1 + 2Z2 + Z3)

2.3.2.3.2.- El Laplaciano

El Laplaciano de una función en 2-D f(x; y) es una derivada de segundo orden definida como:

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

Puede implementarse de varias maneras en forma digital. Para una región de 3 x 3, la forma más frecuentemente encontrada es:

0	1	0
1	-4	1
0	1	0

Máscara de 3 x 3 para el Laplaciano, se aproxima como:

∇f = (Z2 + Z4 + Z6 + Z8) - 4Z5

El requerimiento básico en la definición del Laplaciano digital es que los coeficientes asociados con el píxel central sean positivos y que coeficientes asociados con los píxeles externos sean negativos. Porque el Laplaciano es una derivada, la suma de los coeficientes tiene que ser cero. Por lo tanto la respuesta es cero siempre que el punto tratado y sus vecinos tengan el mismo valor.

El Laplaciano responde a las transiciones en intensidad, y este es raramente usado por la detección de bordes. Porque como es una derivada de segundo

orden el Laplaciano es demasiado sensible al ruido, produce bordes dobles y es incapaz de detectar la dirección del borde. Por lo que el Laplaciano sirve como detector para decir cuando un píxel esta sobre el lado brillante u oscuro de una imagen.

El Laplaciano nos sirve para encontrar la localización de los bordes utilizando la propiedad de cruce por cero y también juega el papel de detectar si un píxel está en el lado claro u oscuro de un borde.

2.3.2.3.3.- Algoritmo

El gradiente y el Laplaciano pueden usarse para formar una imagen en tres niveles, como sigue:

$$s(x,y) = \begin{cases} 0, & \text{si} \quad \nabla f < T \\ +, & \text{si} \quad \nabla f \geq T \quad \text{y} \quad \nabla^2 f \geq 0 \\ -, & \text{si} \quad \nabla f \geq T \quad \text{y} \quad \nabla^2 f < 0 \end{cases}$$

2.3.2.4. - Método de Canny

El algoritmo de Canny es muy eficiente y da un paso más en la detección de bordes. También lo hemos implementado, tiene varias ventajas como conseguir detectar los bordes importantes

Este método se basa en dos criterios fundamentalmente:

- El criterio de localización: Dice que la distancia entre la posición real y la localizada del borde se debe minimizar.

- El criterio de una respuesta que integre las distintas respuestas a un único borde.

Pseudocódigo del algoritmo de Canny:

- Se suaviza la imagen *f(m,n)* con un filtro gaussiano *G(m,n)*, con parámetro de escala σ,

$$g(m,n) = e^{-\frac{m^2+n^2}{2\sigma^2}}$$

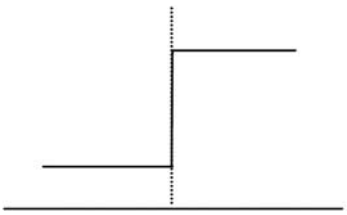
- Para cada píxel de la imagen se estiman las magnitudes del gradiente

$$g(m,n) = \sqrt{D_x^2(m,n) + D_y^2(m,n)}$$

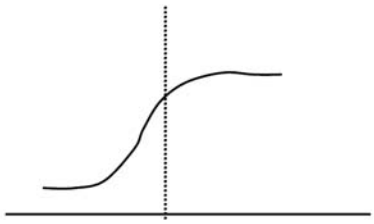
y las direcciones de los bordes locales

$$\alpha(m,n) = \tan^{-1}\left(\frac{D_x(m,n)}{D_y(m,n)}\right)$$

- Se determinan los máximos locales de las magnitudes del gradiente. Cuando los bordes son del tipo escalón, como en la figura (a) de abajo, o rampa, la convolución con una gaussiana presenta los bordes de forma suavizada, y el borde local corresponde al punto de máxima pendiente, es decir, al valor máximo de su derivada. Por lo tanto, la ecuación anterior nos determina ese máximo.



(a) Borde del tipo escalón.



(b) Borde suavizado.

- Supresión no maximal: Se eliminan los bordes de tipo escalón mediante el operador umbral con *histéresis*. Es decir, se seleccionan dos valores umbrales, uno superior y otro inferior. Los píxeles cuyo valor de la magnitud del gradiente está por encima del umbral superior se consideran como partes de bordes y se les llama bordes fuertes, mientras que los que tienen una respuesta por debajo

del umbral inferior no se consideran bordes. Los que están entre ambos valores se les llama bordes débiles.

- Enlace de bordes: El algoritmo incorpora como bordes aquellos borde débiles que están conectados a bordes fuertes, es decir, que están dentro de un entorno 3×3. Se repiten los pasos anteriores para valores ascendentes del parámetro de escala σ.

- Utilizando la información disponible acerca de los bordes para diferentes escalas se aplica la síntesis de características propuesta por Canny. Consiste en marcar primero todos los bordes detectados con el valor más pequeño del parámetro de escala y a partir de ellos se predicen los bordes que se detectarían con un operador para un valor grande de σ (bordes sintetizados). Entonces la respuesta sintetizada de bordes se compara con la respuesta de ella para σ grande. Se marca un borde adicional sólo si su respuesta es más fuerte que la predicha.

2.3.3.- Crecimiento de regiones

El Crecimiento de Regiones consiste en elegir una serie de píxeles iniciales que llamaremos puntos semilla. Después aplicaremos una función a sus vecinos y lo asociaremos a ellos en el caso de que la verifiquen.

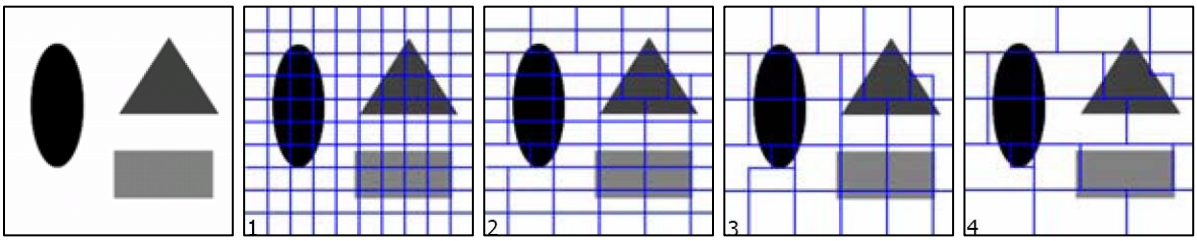
$$|f_{ER}(X_R,Y_R) - f(x,y)| < T$$
$$dist[(x_R, y_R),(x,y)] < D$$

La similitud de las regiones se emplea como criterio principal en las estrategias de crecimiento de regiones. Este criterio lo aplicaremos a cualquier propiedad medible sobre la imagen: nivel de gris, color, textura, forma, modelo, etc. Las técnicas de crecimiento de regiones funcionan, generalmente, mejor en imágenes con ruido donde es muy difícil detectar bordes. Para que el algoritmo de buen resultado debemos elegir bien las semillas y la función de asociación entre píxeles.

Ahora dividimos la imagen en cuadrados del mismo tamaño que llamaremos regiones. Para la unión de regiones se parten de una sobresegmentación de la imagen y se agrupan regiones según el Criterio de Homogeneidad. Los diversos métodos se diferencian en la definición de la segmentación inicial y en el criterio empleado para la unión de regiones. Los resultados obtenidos dependen, además, del orden en el cual se realice la unión de las regiones elementales.

Los métodos más simples comienzan la unión partiendo de una segmentación de la imagen en regiones de 2x2, 4x4 u 8x8 píxeles. A cada región elemental se le asigna una descripción basada en sus estadísticas de determinadas propiedades del nivel de gris. La descripción de cada región se compara con la descripción de sus regiones adyacentes; si ambas coinciden, entonces se unen en una nueva región más grande y se calcula la descripción de la nueva región. La unión de regiones continua entre todas las vecinas hasta que no puede realizarse ninguna nueva unión; en ese momento se obtiene la segmentación.

Gráficamente sería:



La división de regiones es el proceso opuesto a la unión de regiones. Comienza con la imagen representada como una única región, que en general no satisface la condición de homogeneidad. Las regiones existentes en la imagen se dividen secuencialmente para incrementar el cumplimiento de la condición de homogeneidad.

Los métodos de unión y de división emplean, generalmente, criterios de homogeneidad similares y sólo se diferencian en la dirección de su aplicación. La combinación de técnicas de división y unión da lugar a métodos que retienen las ventajas de ambos métodos. Si una región no cumple el criterio de homogeneidad se subdivide en varias regiones. Si varias regiones adyacentes cumplen globalmente el criterio de homogeneidad se unen en una región.

El postprocesado sencillo, empleado con mayor asiduidad, es la reducción de regiones pequeñas en la imagen que no han sido unidas a regiones adyacentes al no cumplir estrictamente los criterios de homogeneidad. El problema es que genera formas con fronteras dadas por el elemento utilizado en la división.

2.3.4.- Watershed

Una forma amable de presentar este método de segmentación es compararla imagen con un terreno en el que la intensidad de cada píxel indica la altura del terreno en ese punto. La segmentación se hace simulando una lluvia sobre la imagen; cuando dos charcos se unen para formar uno más grande, se marcan como bordes los píxeles en los cuales se unieron los dos charcos.
Un problema de este método es la llamada sobresegmentación. El método produce una gran cantidad de regiones pequeñas. Esto se debe principalmente al ruido y a los errores en la cuantización de la imagen [DoTe].

Una de las estrategias usadas para atacar este problema es fusionar regiones similares. En esta variación se usa una aproximación diferente, permitiendo que surja agua sólo en ciertos sitios, denominados semillas. Esta modificación ha mostrado ser bastante efectiva para segmentación de imágenes.

Los resultados de la segmentación usando Watershed con esa modificación pueden verse en la figura de abajo, donde se pintan las semillas para el método en los núcleos de cada célula.

2.3.5.- Transformada de Fourier

Una señal puede ser definida de dos formas: en el dominio de la frecuencia o en el dominio del espacio (para procesamiento de imagen). En el dominio de la frecuencia se utiliza una función *F(U)* donde *u* es una frecuencia de señal y *F(U)* da como resultado un valor con parte real y parte imaginaria, a partir de los cuales se puede deducir la *amplitud* o magnitud y la diferencia de fase de la señal sinusoidal perfecta de frecuencia *u* que forma parte de la señal que representa. Es decir:

$$F(u) = R(u) + I(u) \cdot i$$
$$\left|F(u)\right| = \sqrt{R^2(u) + I^2(u)}$$
$$\Phi(u) = tan^{-1}\left[\frac{I(u)}{R(u)}\right]$$

donde |*F(U)*| y *F* (*U*) constituyen la magnitud y diferencia de fase de la señal sinusoidal perfecta de frecuencia *f(x)* componente de la señal original. Una imagen es el resultado de la composición de muchas señales con diferente frecuencia espacial.

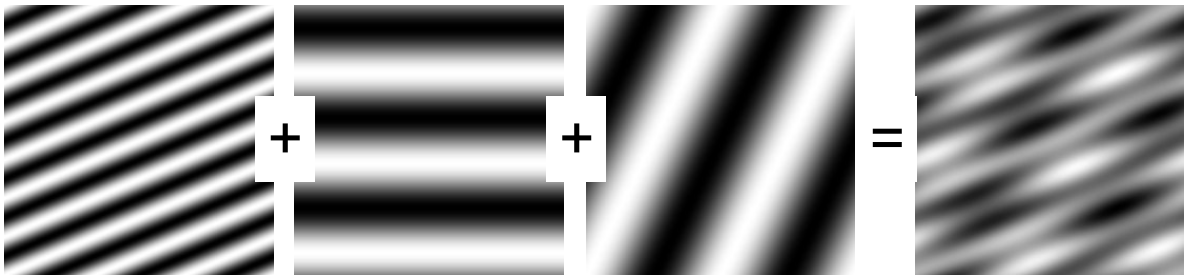
El paso de una señal del dominio de la frecuencia al dominio del espacio y viceversa se puede llevar a cabo a través de las transformaciones de Fourier:

$$F(u) = \int_{-\infty}^{\infty} f(x) \cdot e^{-i2\pi ux} \cdot dx$$

$$f(x) = \int_{-\infty}^{\infty} F(u) \cdot e^{+i2\pi ux} \cdot du$$

Ambos dominios, espacial y frecuencial, son duales. Los dos contienen la misma cantidad de información. La transformación de uno a otro es unívoca.
Cualquier señal se puede expresar como una suma de señales sinusoidales. También en imágenes, pero con señales sinusoidales 2D.

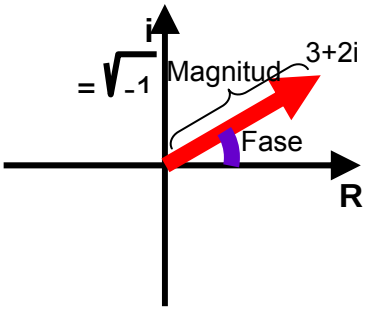
Ejemplo. Imagen como suma de componentes frecuenciales.



En concreto, el valor de cada píxel indica la magnitud y la fase de la componente de frecuencia correspondiente [Giga].

- Magnitud: mayor o menor fuerza del componente.

- Fase: ángulo en el punto 0.Por lo tanto, cada píxel (x, y) se puede expresar como un vector 2D. Se usan números complejos: parte real y parte imaginaria.



El filtrado en el dominio de la frecuencia puede ser muy selectivo, eliminando sólo componentes de frecuencia específicos. El filtrado en el espacio de la frecuencia se lleva a cabo mediante los siguientes pasos:

- Primero, la imagen en el dominio del espacio es transformada al dominio de la frecuencia.

-La imagen en el dominio de la frecuencia es multiplicada por una máscara. Esta máscara tendrá valor 0 en aquellas posiciones que correspondan a frecuencias que se desean eliminar. En otro caso, esta máscara valdrá 1.

- La imagen, resultante de aplicación del filtro, es transformada de nuevo desde el dominio de la frecuencia al dominio del espacio, mostrando la imagen resultante de eliminar las frecuencias que han sido multiplicadas por 0.

2.3.6.- Segmentación piramidal

Hemos visto que es posible obtener características de los objetos a diferentes escalas utilizando filtros para suavizar. Una herramienta poderosa para manipular información de escala son las llamadas pirámides de imágenes.

Se desarrollaron originalmente para aplicaciones de visión por computadora y métodos de compresión. Una pirámide de imágenes es una colección de imágenes con resolución decreciente arregladas en forma de pirámide.

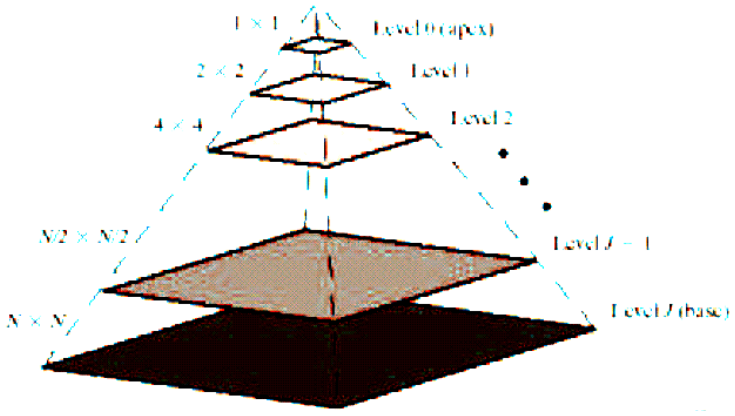
La imagen de la base de la pirámide contiene la más alta resolución, mientras que la punta de la pirámide contiene la más baja resolución. Conforme nos movemos hacia arriba de la pirámide, tanto el tamaño como la resolución de las imágenes disminuyen.

2.3.6.1.- Pirámides Gaussianas

En una pirámide Gaussiana cada nivel de la pirámide es suavizado por un kernel Gaussiano simétrico y submuestrado para obtener la siguiente capa. Notacion: s↓submuestrea la imagen I.

$$P_{Gaussian}(I)_{J-1} = S^{\downarrow}(G_{\sigma} \otimes P_{Gaussian}(I)_{J})$$

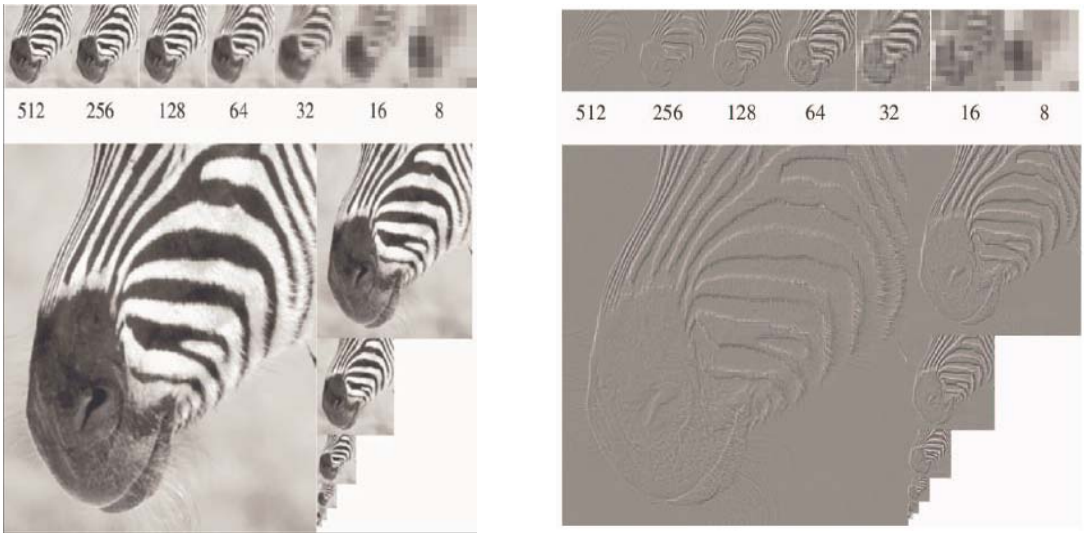
donde, $P_{Gaussian}(I)_{J} = I$; es la original



3.3.6.2.- Pirámides Laplacianas

Las pirámides Laplacianas hacen uso del hecho de que una capa gruesa de la pirámide Gaussiana predice la apariencia de la siguiente capa fina. Si utilizamos un operador de sobremuestreo que pueda producir una versión de capa gruesa del mismo tamaño que la siguiente capa fina, entonces sólo necesitamos almacenar la diferencia entre estas dos predicciones y la siguiente capa fina. Es decir, es una secuencia de imágenes de error, diferencias de dos capas de la pirámide Gaussiana. Cada una de las capas finas de la pirámide Laplaciana es la diferencia entre una capa de pirámide Gaussiana y una predicción obtenida sobremuestreando la siguiente capa Gaussiana de la pirámide.

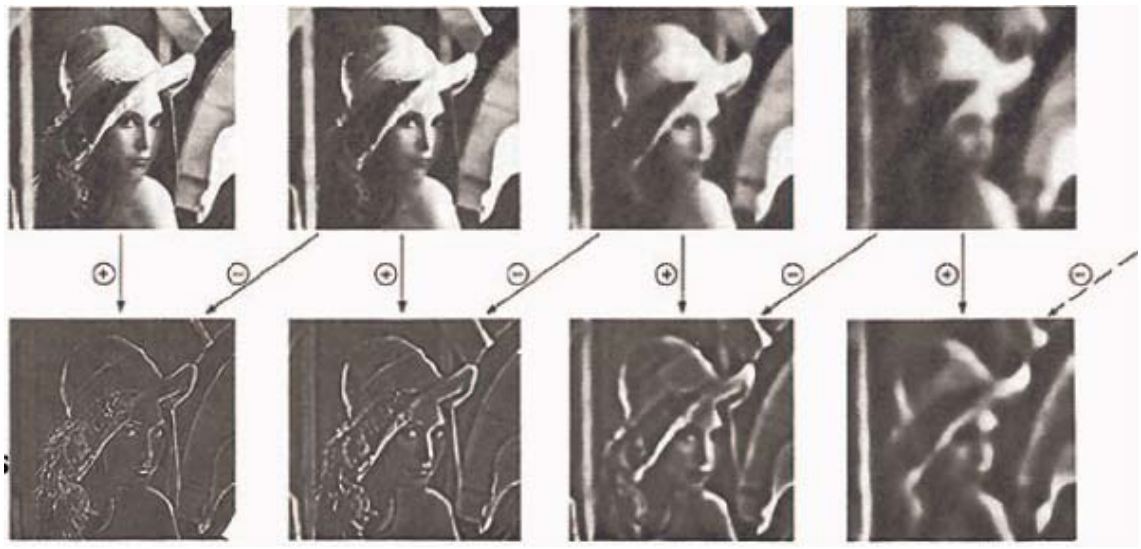
$$P_{Laplaciana}(I)_k = P_{Gaussian}(I)_k - S^{\uparrow}(P_{Gaussian}(I)_{k+1})$$



(a) Pirámide Laplaciana. **(b) Pirámide Gausiana**

En el cado ideal las técnicas que detectan discontinuidades de intensidades deben dar como resultado píxeles que sólo caigan en los bordes (o en las fronteras entre regiones).

Lo que pasa en la vida real es que el conjunto de píxeles detectados rara vez describen un borde de manera completa debido a efectos como: ruido, bordes rotos debido a iluminación no uniforme.



2.3.7.- Transformada de Hough

Los puntos se ligan siempre y cuando se determine si caen o no en una curva de forma específica. El problema es encontrar subconjuntos de n puntos que caigan en líneas rectas.

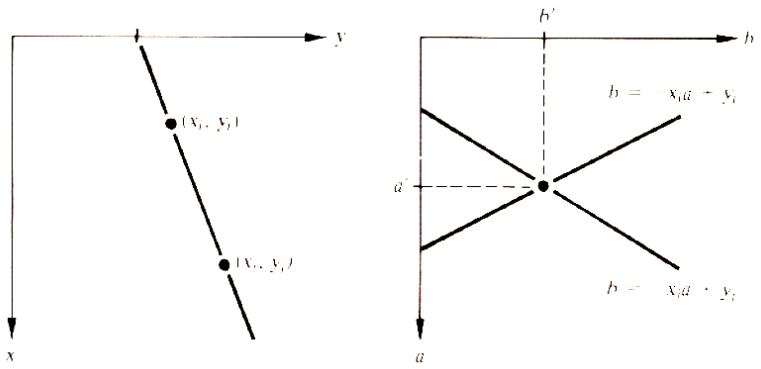
La solución es encontrar todas las líneas determinadas por cada par de puntos y encontrar todos los subconjuntos de puntos cercanos a líneas particulares. Esto involucra hacer:

n(n-1)/2n	líneas
n(n(n-1))/2n	cálculos

para comparar cada par de puntos a todas las líneas lo cual resulta prohibitivo.

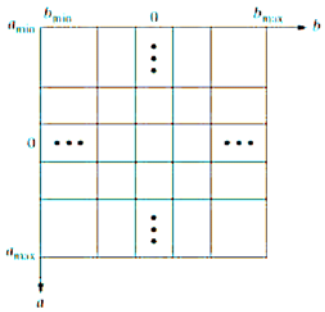
La mejor solución es la transformada de Hough. La ecuación de la línea que pasa por el punto (xi,yi):yi=axi+b; a,b varían y pertenecen a una familia de líneas.

Pero si escribimos: b =-xia +yi ecuación de una sola línea de un par de puntos fijos (xi,yi) en el plano ab, también llamado espacio paramétrico.



Un segundo punto (x_j,y_j) también tiene una línea en el espacio paramétrico asociado a él, con pendiente a‘ y ordenada b‘.

Se subdivide el espacio paramétrico en celdas llamadas acumuladoras, donde (a_{max}, a_{min}) y (b_{max}, b_{min}), son los valores de las pendientes y ordenadas al origen respectivamente. La celda de coordenadas (i,j), con valor de acumulador A(i,j), corresponde al cuadro asociado con las coordenadas (a_i,b_j) del espacio paramétrico. Se inicializa en cero.

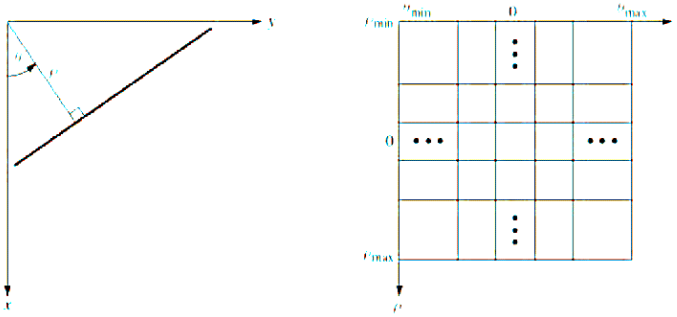


Para cada punto (x_k,y_k) del plano de la imagen, dejamos al parámetro *a* tomar cada uno de los valores de las subdivisiones del eje-a del espacio paramétrico y se resuelve la ecuación b = - x_k a + y_k.
Los valores de b resultantes se redondean al valor más próximo en el eje-b. Si una elección a_p resulta en una solución b_p, entonces incrementaremos la celda

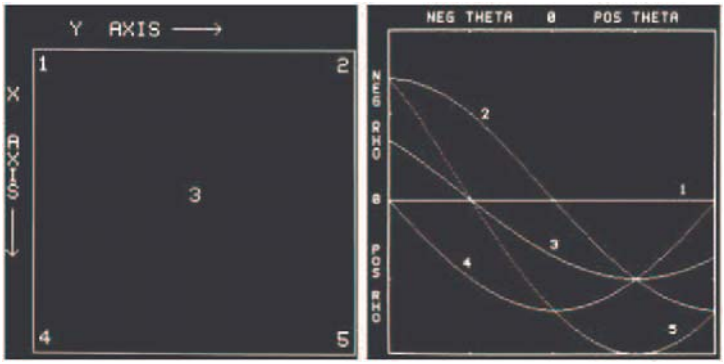
A(p,q)=A(p,q)+1. Al final del procedimiento el valor Q en A(i,j) corresponderá a Q puntos en el plano-xy que caen en la recta y =aix +bj. El número de divisiones en el *plano-ab* determina la exactitud de la colinearidad de dichos puntos.

Mientras la pendiente *a*, se aproxima al infinito la recta se aproxima a la vertical, por lo tanto, podemos usar la representación normal de una línea.

Aunque existe un problema al utilizar la ecuación de la recta: y = aix +bj. La pendiente a se aproximan al infinito la recta se aproxima a la vertical, por lo tanto, podemos usar la representación normal de una línea: xcosθ + ysin =*p.*



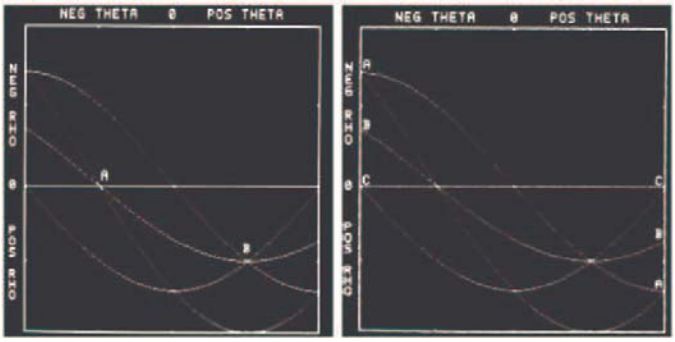
Para valores de *θ* se rsuelve *p.*



(a) **(b)**

(a) 5 puntos en la imagen

(b) 5 puntos mapeados plano (*p,θ*)



(a) **(b)**

(a) A-denota intersección entre puntos 1,3,5 en el plano xy, (*θ* =-45º)

(b) B-denota intersección entre puntos 2,3,4 en el plano xy, recta (*θ* =45º)

CAPÍTULO 3

DISEÑO Y RESOLUCIÓN DEL TRABAJO

3.1.- Introducción

En este proyecto se evaluar la posible utilidad de las ontologías en el mundo del procesamiento de imágenes. En primer lugar hemos implementado diferentes métodos de clasificación para poder diferenciar imágenes de diez dominios.

Los dominios elegidos (ver Anexo) son:

- Primeros planos de caras humanas.
- Girasoles con los pétalos amarillos.
- Banderas de distintos países.
- Radiografías de cerebros.
- Radiografías de algunas partes del cuerpo.
- Campos de golf.
- Algunas imágenes que tuvieran el escudo de Ferrari.
- Tigres en cualquier lugar.
- Desierto de día.
- Distintos tipos de planos con algunas líneas.

Las imágenes se han elegido por sugerencia de los directores del proyecto aunque se ha tenido en cuenta que fuesen adecuadas para los distintos tipos de extractores de características.

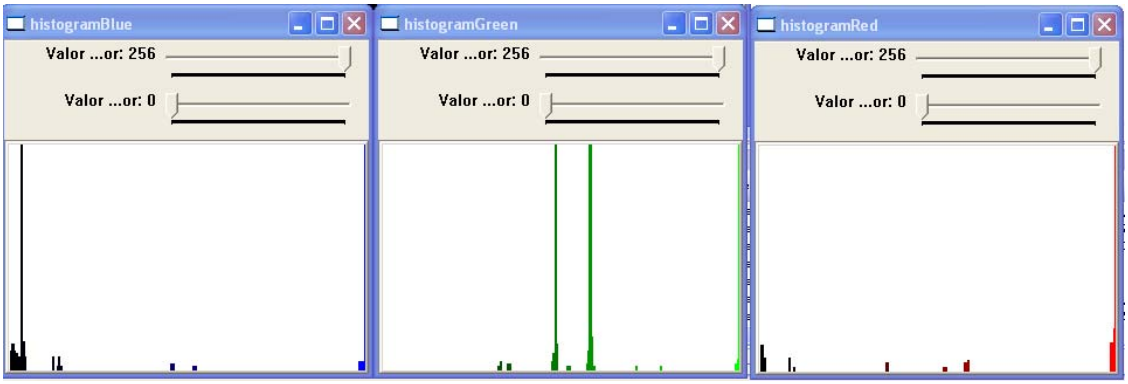
También hemos implementado algunos métodos de segmentación y hemos probado sus resultados. A continuación hemos desarrollado una aplicación que utiliza ontologías para ayudar a elegir un método de segmentación y un método de clasificación.

3.2.- Extracción de características

En un principio vamos a tener 10 dominios, por lo que tendremos imágenes de distintos dominios como entrada y como salida nos dirá a qué dominio pertenecen. Para determinar a qué dominio pertenece nos vamos a basar en algunas técnicas o algoritmos propios pertenecientes a los distintos dominios.

3.2.1- Número de puntos en el histograma

Las banderas se diferencian del resto de las imágenes en que poseen poca diversidad de colores. De esta manera, al hacer el histograma, todos se acumulan en los mismos puntos. En la imagen ponemos un ejemplo:



Nos damos cuenta de que la mayoría de los píxeles tienen los mismos valores. Simplemente debemos hacer el histograma y consultar los valores para los que no es cero. Si este número no supera cierto umbral diremos que es una bandera.

3.2.2 - Color de puntos

Este método es sumamente sencillo pero nos permite diferenciar unas imágenes de otras. La idea de este método es localizar en las imágenes un determinado tipo de color para discernirlas del resto.

Los girasoles tienen un color amarillo muy particular este es un factor determinante que nos permite diferenciarlo del resto de las imágenes. Nos vamos a servir de esta característica para poder diferenciarlo del resto de las imágenes.

Para localizar las imágenes de girasoles nos vamos a servir de un método que consta de dos partes:

Parte 1

Este algoritmo tiene como entrada un conjunto de imágenes muy amplio del orden de 170, sobre distintos dominios. A continuación lo que queremos es crearnos una aplicación en la que vayan apareciendo imágenes. En el caso de que sea una imagen de girasol, mediante el ratón pincharemos en aquellos puntos donde este el amarillo. Puesto que es el color que queremos analizar. Por lo tanto en el caso de que sea una imagen de girasol, seleccionaremos los puntos de los pétalos que es donde esta el amarillo de los girasoles. En el caso de que sea otro tipo de imagen sencillamente pasaremos a la siguiente. Mediante este sencillo procedimiento ya hemos conseguido almacenar alrededor de 93 muestras que las volcaremos sobre un fichero para usarlas en el siguiente programa.

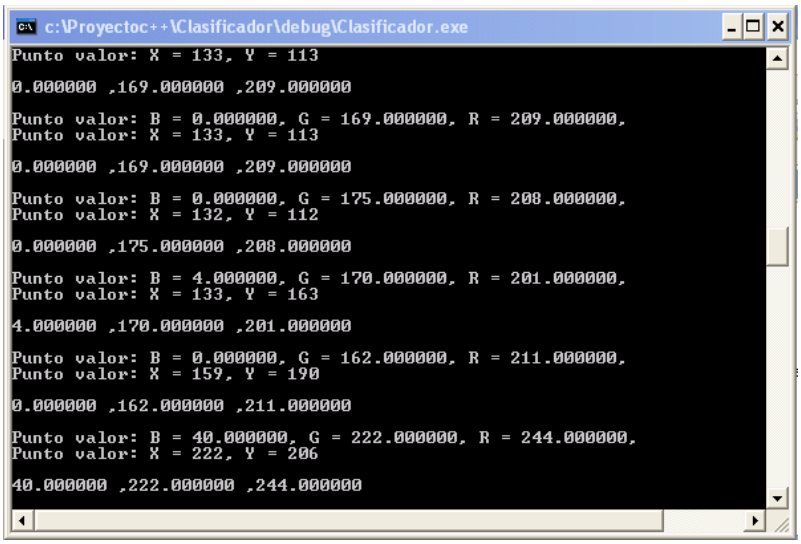
En esta figura mostramos como funciona nuestra aplicación a la derecha podemos observar el girasol y a la izquierda vemos los puntos seleccionados y el valor que poseen. El valor esta dado en función de BGR, esto quiere decir que en primer lugar daremos las coordenadas del Blue, el Green y el Red, es decir, Azul, Verde y Rojo. Cada componente posee un valor de 0 a 255

Valor Punto: B, G, R.

Coordenadas: X, Y.

P(X, Y) : = BGR

Cuando pensamos que hemos tomado las suficientes muestras pasaremos a la siguiente imagen simplemente pulsando una tecla.



Parte 2

Lo primero que haremos será leer el fichero donde se han almacenado los puntos y posteriormente calcularemos la media y la varianza.
La media la calculamos de cada valor por separado y la varianza la calculamos restando la media al valor de cada uno de los puntos. La varianza será la suma de las diferencias de todas las imágenes respecto a la media dividido entre el número de imágenes.

Una vez tenemos la media y la varianza recorremos todas las imágenes y para cada píxel aplicamos la siguiente condición:
Si (componente B – mediaB) < Varianza B y (componente G- mediaG) < Varianza G y (componente R – mediar) < Varianza R, entonces incrementaremos un contador de elementos amarillos.

Para concluir dividiremos el número de elementos entre el número total de píxeles de la imagen por lo tanto veremos el porcentaje de píxeles de girasol de la imagen los resultados son:

Después de calcular la media, calculamos la varianza y esto consiste en ver la diferencia de cada uno de los puntos.

Una vez hemos calculado la media y la varianza, analizamos las imágenes. El análisis consiste en recorrer la imagen, elemento a elemento y para cada elemento lo comparamos con la media y la varianza. De esta manera por cada elemento de la matriz que representa la imagen en los tres niveles RGB, y después de esto. En caso positivo, contabilizamos el número de veces que si y al final de la matriz, dividimos el número de aciertos

La media y la varianza para cada canal son:

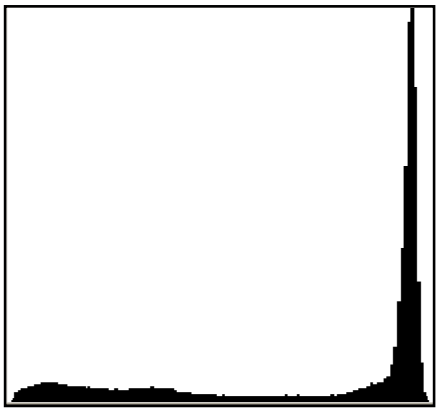
Media B: 12.913979, Media G: 177.537628, Media R: 230.892471.
Varianza B: 13.874199, Varianza G: 32.349865, Varianza R: 18.791538.

Lo que hacemos ahora es analizar los resultados y vemos que para las imágenes de girasoles los valores oscilan entre 25% hasta el 4% y que el resto de las imágenes están entre valores del 0% al 1%.

Podemos concluir que a pesar de ser sencillo este método nos ha dado un buen resultado.

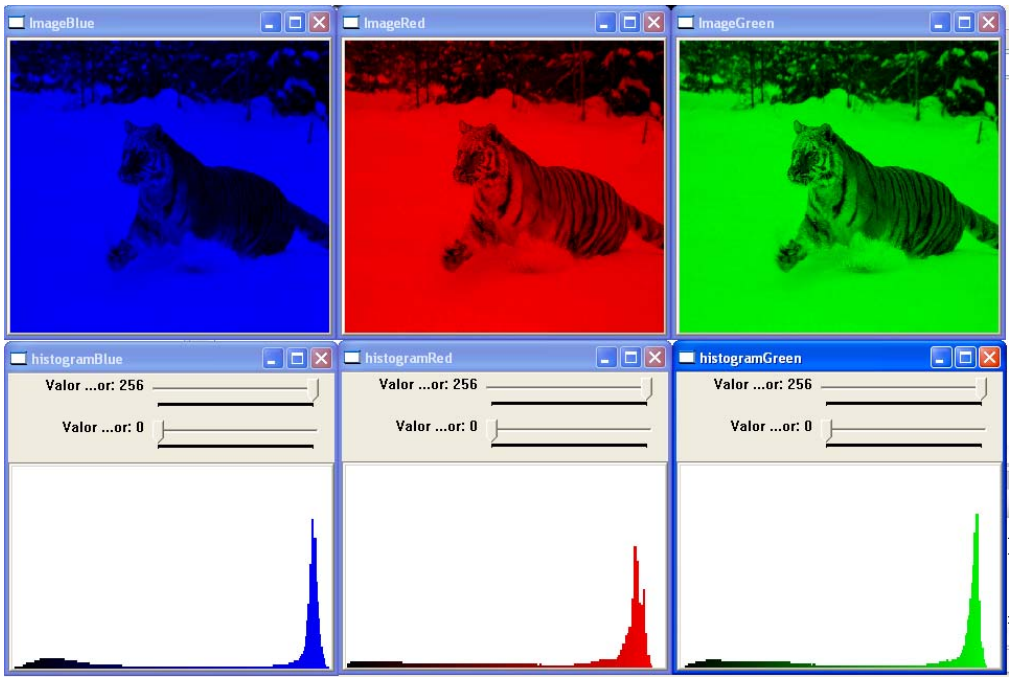
3.2.3.- Valor de los histogramas

En un histograma tenemos dos ejes. Tenemos el eje X y el eje Y. En el eje X tenemos el rango que será los posibles valores que pueden adoptar las variables. En el eje Y se muestran el número de variables que poseen este valor.
A modo de ejemplo vamos a poner un histograma:



La imagen va de 0 a 255. Nos damos cuenta de que en esta imagen en nivel de grises tiene el siguiente histograma.

Antes de hacer el histograma vamos a separar a la imagen en tres imágenes. Es decir dada la imagen A, la dividiremos en el color Rojo, el Verde y el Azul. De esta manera hemos conseguido hacer el histograma de cada color por separado.

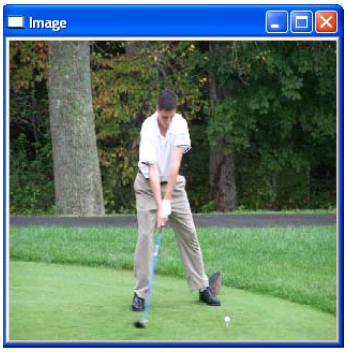


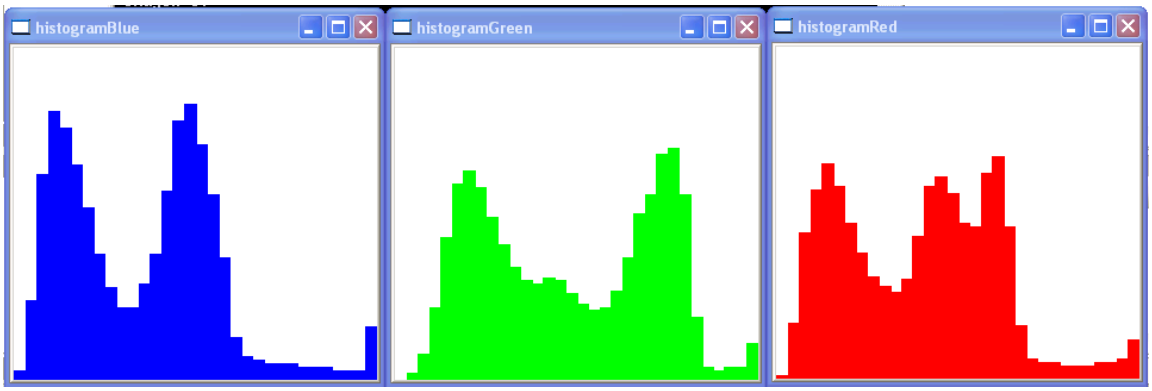
Aquí podemos apreciar el histograma de cada imagen. La imagen del tigre está compuesta por distintos valores de 0 a 255 vemos que en esta imagen se agrupan valores más a la derecha esto es debido a que la imagen posee muchos valores de baja intensidad. Para cada uno de los canales RGB que componen la imagen hemos hecho el histograma, aunque en este método los histogramas se agrupan en cocientes de 32.

Para buscar dentro de una cantidad grande de imágenes las que están más relacionadas podemos usar un procedimiento bastante eficaz. Aquí nos vamos a basar únicamente en los colores y en su cantidad.

El proceso consiste en usar histogramas de dos dimensiones de los tres canales de los canales (R, G, B), con 32 celdas por dimensión.
Una vez hemos hecho el histograma lo que debemos hacer es normalizar los histogramas para que la suma total sea 1.
Después de esto tomamos como medida de similitud el solapamiento entre histogramas. Esta medida irá entre 0 y 1 (entre 0% y 100% de solapamiento) por cada canal.

Ejemplo de los tres histogramas con rango 32.





Tenemos que calcular los histogramas de todas las imágenes de la BD y los de la imagen de A. Después calculamos la similitud de estos histogramas con la imagen A.

Para concluir el resultado son las imágenes de la BD que tengan una mayor similitud.

El pseudocódigo seria:

Para todas las imágenes

Dividirlas en tres planos

CalcularHistograma Plano B, Plano G, Plano R.

Para todas las distintas a la escogida

Dividir en tres planos

CalcularHistograma Plano B, Plano G, Plano R.

Solapamiento B = Comparar Histograma(PlanoB,PlanoB)

Hacemos lo mismo para los otros dos planos.

Si ((SolapamientoB + SolapamientoG + SolapamientoR) < Umbral)

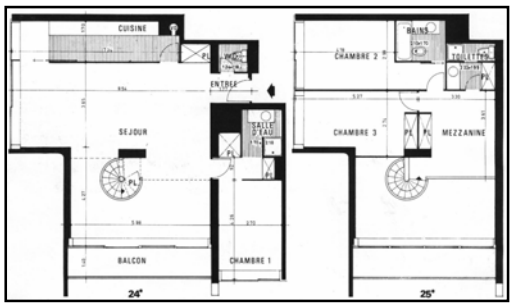
Entonces Imagen A esta asociado con Imagen B

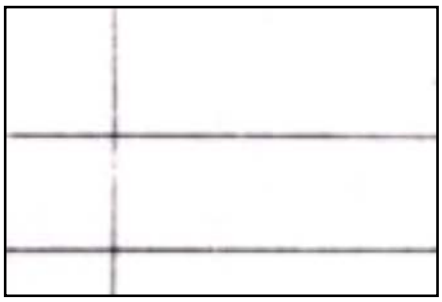
3.2.4.- Número de líneas

Nos hemos dado cuenta de que los planos se distinguen por la presencia de líneas. Para cada línea lo que hacemos es intentar encontrar píxeles consecutivos del mismo color.

La idea es que los pixels de arriba sean de un color los de abajo del todo sean del mismo color y que los de en medio sean todos iguales entre ellos y distintos a los de los bordes.

De esta manera recorremos toda la imagen y por cada coincidencia incrementamos el contador.





En este archivo aparecen en la figura de la izquierda un plano de una obra de una casa. Lo que caracteriza normalmente a un plano es la presencia de líneas. Por lo tanto lo que vamos a tratar de identificar es la presencia de líneas. Una línea la detectaremos de la siguiente forma:
Suponemos que tenemos una zona ABA. La primera B representa la línea y las dos letras A las fronteras de la línea, que serán dos zonas iguales. Al ser dos zonas iguales y la del media igual la podemos contar como línea. Lo que también haremos será hacer el grosor en función de una variable. Esta variable variará de 2 a 11 y es el número de pixels que puede tener la línea de grosor. Lógicamente

todos estos píxeles deberán variar muy poco su color, pero tampoco podemos exigir que sean exactamente iguales porque las líneas no tienen exactamente el mismo color en la realidad. Además de esto debemos tener en cuenta los píxeles de al lado de estas por lo tanto dada pasaremos una máscaras por toda la imagen como se muestra a continuación. En los puntos en los que coincida se contabilizarán y suponemos que para las imágenes que representen planos el número de puntos será bastante mayor.

A continuación ilustramos unas imágenes en las que se puede apreciar en qué consiste el algoritmo realmente.

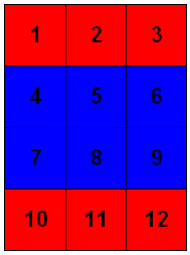
172	29	116	17	3	4	240	190	.....	186
147	47	149	153	136	99	156	170	.....	147
206	178	118	72	109	193	110	223	.....	81
57	253	71	139	19	173	205	21	.....	9
188	51	101	90	43	154	38	243	.....	14
60	195	201	176	98	148	62	208	.....	95
209	82	110	173	31	238	131	72	.....	25
.....	.....	.....	.....	.....	.....	.....	.....	.....	153
89	215	237	179	39	47	136	196	219	229

7	197	143	161	7	62	54	87	.....	245
234	55	187	144	37	227	99	185	.....	134
242	65	115	238	172	227	126	145	.....	37
120	34	64	62	131	90	89	231	.....	101
15	21	231	196	201	250	44	25	.....	40
120	129	48	230	71	190	138	28	.....	67
145	238	87	182	26	50	246	72	.....	89
.....	.....	.....	.....	.....	.....	.....	.....	.....	23
106	206	77	221	89	1	115	16	193	236

Esta imagen nos servirá para ilustrar el algoritmo que queremos crear. La idea del algoritmo es crearse una máscara en el dibujo sería la figura que vemos. La máscara tiene el borde superior y el borde inferior. Una de las condiciones es que todos los píxeles que forman el borde estén formados por píxeles que sean de los mismos colores.

Suponemos que todos los píxeles de la línea son iguales y que la línea puede tener ser de distintos píxeles. Por esa razón hacemos variar el valor de los píxeles del medio que pueden abarcar de 2 a 9.

También repetimos la operación para detectar las líneas verticales. El algoritmo se podría mejorar intentando detectar las líneas en diagonal.



Simplemente debemos comprobar que los píxeles 1, 2, 3 y 10, 11, 12 coincidan en color con cierto margen de error y a su vez coincidan el 4, 5, 6, 7, 8 y 9. Estos píxeles deben ser distintos al primer grupo.

3.2.5- Detección de caras

La detección de caras es una técnica que se emplea con muchísima frecuencia en la visión por computador.

En primer lugar en el caso de que tengamos espacios de altas dimensiones, esto puede requerir una gran cantidad de imágenes porque se presta a distintas variaciones. En el caso de que trabajemos a baja resolución será fácil mostrar un patrón en el que se vean las cejas, boca y ojos. En caso de que nos movamos a alta resolución es difícil representar un patrón que no sea susceptible de pequeñas variaciones [InMa].

Para detectar las caras en las imágenes nos vamos a servir de los una función la llamada cvHaarDetectObjects(); pasándole como parámetro la cascada : haarcascade_frontalface_alt2.xml

El documento “haarcascade_frontalface_alt2.xml” fue diseñado con el objetivo de encontrar caras en las imágenes de una manera rápida. Se basa en tres principios, el primero es la imagen integral, esta permite analizar las características de la imagen muy rápidamente para que sea procesada. También se usan un algoritmo de aprendizaje muy eficiente basado en AdaBoost, este selecciona un número pequeño de características visuales y clasificadores muy eficientes. La tercera aportación es el descarte de algunas regiones por una combinación de clasificación de cascadas que son rápidamente descartadas, mientras que se centra en las regiones más prometedoras.

Esta función encuentra en regiones rectangulares en las que la cascada ha sido preparada y devuelve una lista con los rectángulos que las contienen. La función escanea la imagen a distintas escalas. También aplica una heurística basada en la poda de Canny, por defecto tiene de escala 1.1 y el mínimo de vecinos es tres.

3.2.6- Número de franjas

El algoritmo de clasificación de tigres es un tanto más complejo. El algoritmo tiene dos fases:

Fase 1

En esta fase nos creamos una aplicación en la que se almacenarán los valores de los distintos puntos. Los tigres se distinguen por tener principalmente tres colores. El blanco, el negro y el naranja. De esta forma tomaremos un número importante de muestras para cada tipo de color. Tendremos una estructura para almacenar las muestras de blanco, negro y naranja. Posteriormente cuando veamos que el número de muestras es suficientemente representativo daremos por concluida la fase 1, no sin antes volcar los resultados sobre un fichero.

Fase 2

Lo primero que hacemos en la fase 2 es leer los ficheros y almacenar los resultados en un vector. Posteriormente nos creamos una máscara que inicialmente pondremos todos las píxeles a un mismo valor. Una vez tenemos la máscara recorremos la imagen original del tigre. Para cada punto lo comparamos con las muestras obtenidas para cada valor. Por ejemplo dado un punto P(X,Y) = BGR(145,123,67) lo restamos en valor absoluto con todas las muestras de Naranja. En el caso de que el valor sea menor a un umbral para los tres componentes concluiremos que el píxel pertenece al valor naranja y lo pintaremos en la máscara de ese color.

A continuación mostramos unas imágenes para que se entienda mejor.

Fase 3

Esta fase consiste en analizar las máscaras. Nos hemos dado cuenta de que los tigres poseen valores naranjas alrededor del negro. Vamos a tratar de localizar en esta fase el número de puntos blancos. El numero de píxeles negros rodeados por píxeles naranjas.

Para implementar si un píxel está rodeado la hacemos de la siguiente manera:

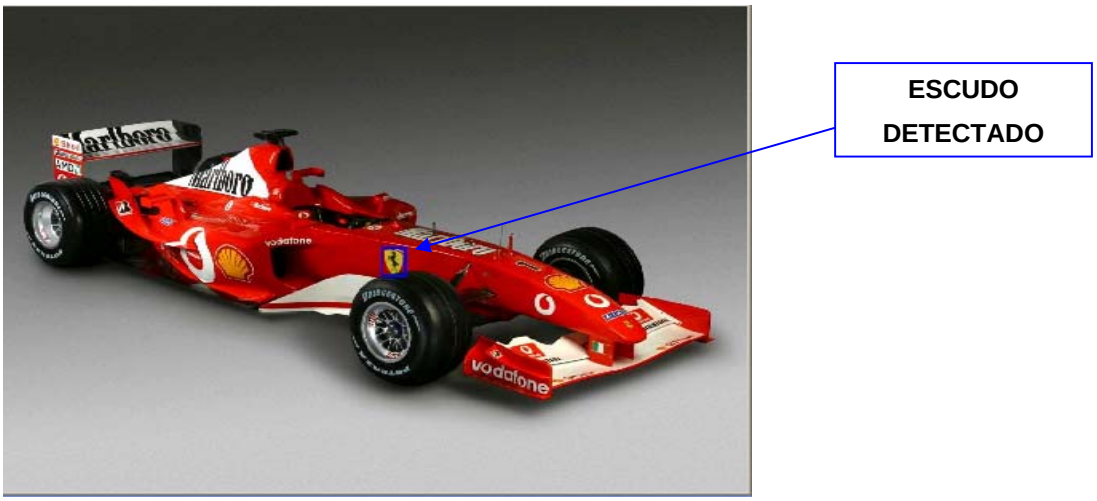
Para todos los puntos de la imagen.
Si el punto es esta entre la columna tercera y la del final menos tres.
Si el píxel es de color rosa.
Si tiene tres píxels seguidos naranjas por la derecha e izquierda.
Entonces Incrementamos el numero de puntos.

Una vez aplicamos esto a todas las imágenes nos damos cuenta de que las imágenes que más puntos tienen son las de los tigres.

3.2.7- Template-matching

Para encontrar imágenes de Ferrari vamos a emplear la técnica de la búsqueda de patrones. En este tipo de imágenes buscaremos el escudo de Ferrari dentro de la imagen. En el caso de que no encontremos ningún escudo concluiremos que la imagen no es de Ferrari.Para aplicar este método lo que debemos hacer es buscar un patrón. En este caso el patrón será el escudo que se muestra abajo.

La imagen que ponemos tiene una calidad baja para que se adapte mejor a las posibles variaciones que puede haber en las distintas imágenes. La idea es pasar la imagen como si fuera una máscara e ir comparándola con el resto de los píxeles. Cuando haya una coincidencia superior a un umbral concluiremos que se que se encuentra. Debido a que el escudo puede estar en distintos tamaños lo que haremos es aumentar y disminuir el tamaño del patrón, es decir, hacer un barrido de las distintas escalas.

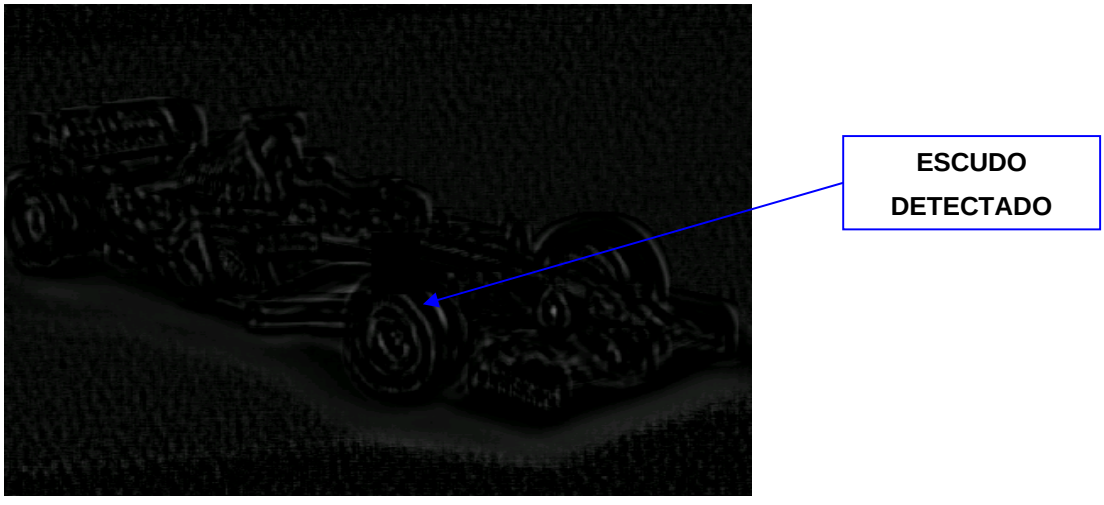


Vemos como el algoritmo ha seleccionado mediante un cuadrado azul la imagen de Ferrari.

La idea de la técnica consiste en encontrar un patrón dentro de una imagen mayor. Lógicamente no buscamos que el patrón se adapte al cien por cien al de la imagen, ya que en este caso esta técnica funcionaria en contadas excepciones. Vamos a permitir cierto margen de error; esto nos dará la ventaja de que será más fácil encontrar el patrón que buscamos pero puede pasar que confunda el patrón con una imagen parecida. Para implementar esta técnica nos vamos a servir de Template Matching.

Dada una imagen A de tamaño (WxH), (estas siglas vienen del inglés height alto y width ancho) y sea un patrón P de dimensiones (wxh). El resultado de esta técnica es una imagen M con un tamaño de (W-w+1)x(H-h+1).

La Matriz M contiene la probabilidad de que el rectángulo [x,y] – [x+w-1, y+h-1] de A tenga el patrón P [Giga].



El valor de cada píxel viene dado por el valor de alguna función de cálculo, que en nuestro caso será la suma de diferencias

$$M(x,y):= d(\{A(x,y),...,A(x+w-1,y+h-1)\}, \{P(0,0),..., P(w-1,h-1)\})$$

La suma de diferencias al cuadrado se representa:

$$M(x, y):= \sum_{a=0..w-1} \sum_{b=0..h-1} (P(a, b) - A(x+a, y+b))2$$

La imagen M también se puede representar. Debemos normalizar los píxeles para que todos estén en el rango 0..255.

La normalización se puede realizar mediante esta sencilla operación:

$$sqrt(\sum_{a=0..w-1} \sum_{b=0..h-1} P(a, b)2 \cdot \sum_{a=0..w-1} \sum_{b=0..h-1} A(x+a, y+b)2)$$

Los valores están comprendidos entre -1 y 1, para que esté en el rango debemos sumarle 1 y multiplicarlo por 255. Cuanto más cercano este a 255 más probabilidad tendrá.

Ahora tenemos una matriz que se puede representar en escala de grises. Los valores oscuros indican que el patrón tiene una probabilidad alta y los valores blancos indican lo contrario.

Pseudo-código el algoritmo empleado:

Conseguir un patrón de lo que queremos encontrar.
Aplicar el template matching a la imagen..

Buscar los máximos locales de M.
Buscar el máximo absoluto, (lx, ly)= argmax ∀x, y M(x, y).
Si M(lx, ly) es menor que cierto umbral, acabar.
Añadir la posición (lx, ly) a una lista de localizaciones resultantes del proceso.
Poner a cero en M el rectángulo [lx-w,ly-h]–[lx+w,ly+h].
Volver a buscar el máximo absoluto.

Antes de concluir debemos tener en cuenta dos detalles. Para poder perfeccionar esta técnica es necesario aplicar varias escalas, ya que el patrón puede estar en varios tamaños en las distintas imágenes. Además de varias escalas podemos hacer algunas rotaciones porque la imagen puede enfocarse desde otros ángulos.

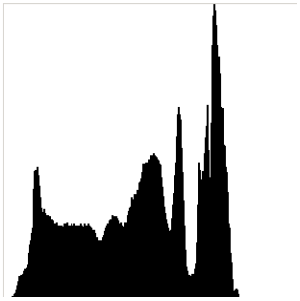
3.3.- Segmentación

Los resultados obtenidos con los distintos algoritmos de los que hicimos una descripción en el punto 2.3.

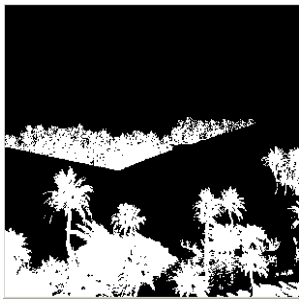
3.3.1. – Umbralización

3.3.1.1. – Media

(a)Imagen desierto



(b) Histograma desierto



(c) Binarización desierto

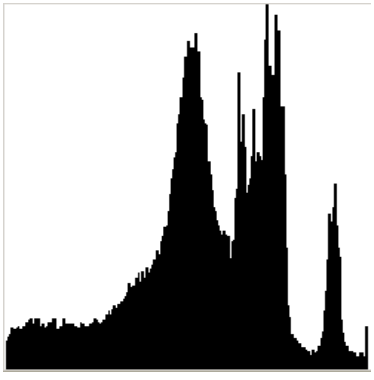
En la primera imagen tenemos un desierto codificado en escala de grises. La segunda representa el histograma y la tercera representa el resultado de binarizar, es decir, poner los puntos inferiores a un umbral a blanco y a los superiores a negro.
Para probar el algoritmo lo hemos implementado y nos hemos servido de las funciones del Opencv de:

F1 = cvLoadImage();
F2 = cvGet2D();
F3 = cvSet2D();

En este algoritmo lo que hemos hecho es cargar la imagen con la función F1, después recorremos todos los píxeles. Mediante la función F2 leemos su valor y se lo sumaremos a una variable que nos servirá para calcular la media de todos los píxeles. Volvemos a leer el valor de los píxeles. Usaremos la F3, para poner en blanco si el número es menor y negro si es mayor a un umbral.

3.3.1.2.- Búsqueda del Máximo y Mínimo

(a) Imagen de golf



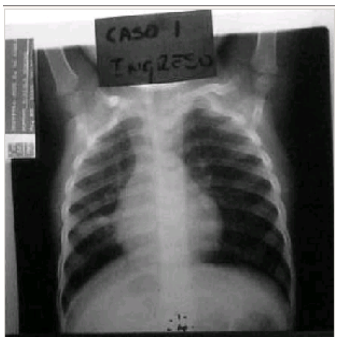
(b) Histograma de golf

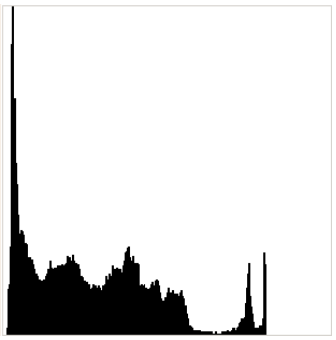
(c) Binarización

Hemos detectado el máximo del histograma en la posición 185 con el valor 200, el segundo máximo lo hemos encontrado en la posición 129 con el valor 180 y el mínimo entre estos dos en la posición 159 con el valor 58. Por lo que hemos puesto en blanco los valores por debajo de 159 y en negro los superiores.

3.3.1.3. - Algoritmo de Otsu

Los resultados obtenidos en nuestra aplicación son los siguientes:
Valor calculado como umbral 108, gradiente máximo 255 y mínimo 0.





(a) Esqueleto **(b) Histograma esqueleto** **(c) Binarización**

El algoritmo de Otsu es el más complejo de todos pero también es el que mejores resultados ha dado. Su implementación es similar al anterior pero calculando el umbral de una manera diferente [UmbSeg].

Pseudocódigo de la función de Otsu:

Calculamos el valor del píxel mayor y del menor.
Hacemos el histograma de la imagen.
*Calculamos la Csum como = x*F(x) para x = 0 hasta 255, Donde x es el índice y F(x) su valor.*
Calculamos N = F(x) para x = 0 hasta 255.

Para todos los valores del histograma

N1 = F(x) Sumatorio desde cero hasta el índice actual

N2 = N - N1

*Sum = x*f(x) sumatorio valor histograma de cero hasta el indice actual*

M1 = Csum/N1

M2 = (Sum-Csum)/N2

*Sb = N1*N2*(M1 - M2)*(M1 - M2)*

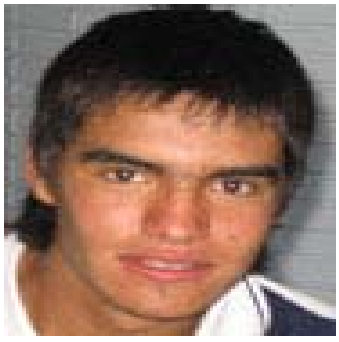
Nos quedamos con el índice del mayor Sb.

3.3.1.4.- Detector de piel

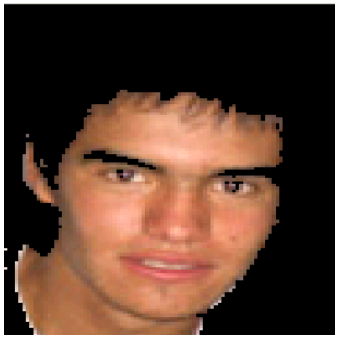
Los criterios serán:

1. $R > 95 \wedge B > 20 \wedge G > 40$
2. $\text{Max}\{R, G, B\} - \text{Min}\{R, G, B\} > 15$
3. $|R - G| > 15$ and $R > G$ and $R > B$

Y los resultados son:



(a) Foto Carnet



(b) Detección piel

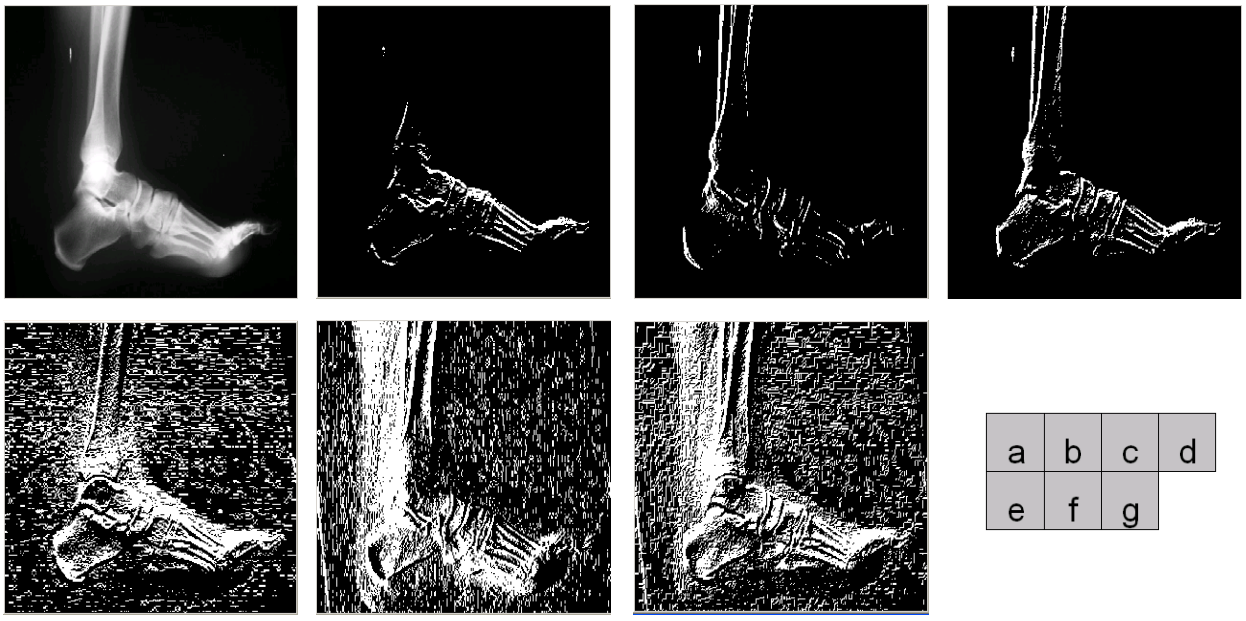
(c) Binarización

En primer lugar detectamos los píxeles que son de piel en la imagen (a). Esto se refleja en la imagen (b). En la imagen (c) vemos que hemos binarizado la imagen en función de si el píxel pertenece o no a la imagen.

3.3.2.- Detección de discontinuidades

3.3.2.1. – Detección de puntos, líneas y bordes

Resultados obtenidos con las máscaras de Sobel:



a	b	c	d
e	f	g	

En la imagen (a) mostramos la imagen original. En la (b) se muestra la imagen original pero aplicandole la máscara de sobel del eje horizontal y después binarizandola. La imagen (c) es el resultado de aplicarle la máscara de sobel en vertical a la imagen original y binarizarla. (d) La imagen (d) es simplemente la suma de la imagen b y la c. En la imagen (e) se muestra el resultado de aplicarle a

la figura (a) la máscara de en horizontal Sobel pero sin binarizar. La imagen (f) es igual que la (e) cambiando la máscara a vertical. En la imagen (g) se muestra el resultado de sumar la imagen (e) con la (f).

Este algoritmo ha sido fácil de implementar porque la librería OpenCV tiene funciones que permiten realizar las operaciones antes mencionadas. Estas funciones son:

```
F1 = cvSmooth(suavizada, suavizada, CV_GAUSSIAN, 5, 5);
F2 = cvThreshold( suavizada, suavizada, 125, 255.0, CV_THRESH_BINARY);
F3 = cvSobel( entrada, PlanoY, 1, 0, 3);
```

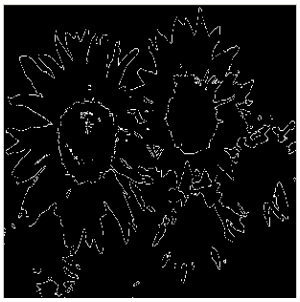
Con la función F1 suavizamos la imagen que leemos. Con la función F3, calculamos el gradiente en la dirección X o Y en función de los parámetros que le pasemos. La función F2 nos sirve para binarizar la imagen.

Además del método de Sobel hemos usado el de Laplace. Los resultados han sido los siguientes:

(a) Flor

(b) Suavizado

(c) Binarización



(d) Paso cero

```
F1 = cvSmooth(suavizada, suavizada, CV_GAUSSIAN, 5, 5);
F2 = cvThreshold( suavizada, suavizada, 125, 255.0, CV_THRESH_BINARY);
F3 = cvLaplace( suavizada, pasoscero, 3);
```

La imagen (a) es una imagen de unos girasoles codificada en nivel de grises. Posteriormente la aplicamos a esa imagen la función F1 y realizamos el suavizado (Imagen b) para evitar cambios fuertes en los píxeles. Al final realizamos una umbralización en la imagen (c) mediante la función F2. La imagen (d) la obtenemos al aplicarle la función F3 a la imagen anterior.

3.3.2.2. - Método de Canny

(a) Imagen cara

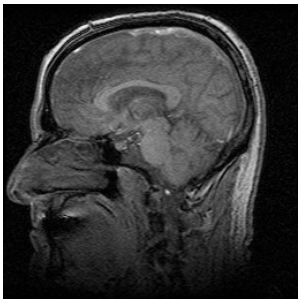
(b) Método de Canny

```
F1 = cvSmooth( gray, edge, CV_BLUR, 3, 3, 0, 0 );
F2 = cvNot( gray, edge );
F3 = cvCanny(gray, edge, (float)edge_thresh, (float)edge_thresh*3, 3);
```

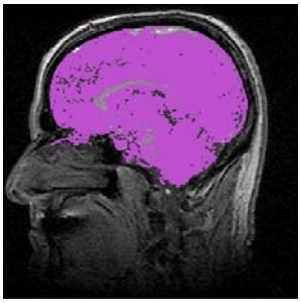
Con la función F1 suavizamos la imagen que nos pasan. Después hacemos el negativo con la función F2. El algoritmo de Canny lo aplicaremos con la función F3.

3.3.3.- Crecimiento de regiones

Estos son los resultados:



(a) Imagen Cerebro



(b) Crecimiento de regiones

F1=cvFloodFill(gray_img, seed, brightness, cvRealScalar(lo),cvRealScalar(up), &comp, flags, is_mask ? mask : NULL);

Esta función nos permite aplicar el algoritmo antes citado. Este algoritmo permite establecer varias semillas, pero para el reconocimiento de cerebros nos ha sido suficiente con una semilla.

3.3.4.- Watershed

(a) Imagen Golf

(b) Transformada Watershed

F1 = cvWatershed(img0, markers);
Nos hemos servido de la función F1 para aplicar nuestro algoritmo.

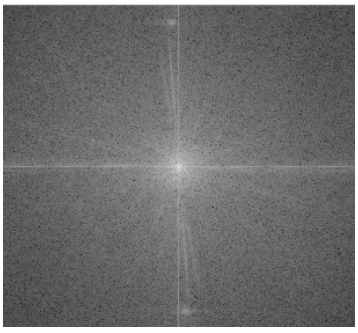
3.3.5.- Transformada de Fourier

Este es el resultado:

F1 = cvGetOptimalDFTSize(im->height - 1);
F2 = cvDFT(dft_A, dft_A, CV_DXT_FORWARD, complexInput->height);
F3 = cvLog(image_Re, image_Re); // log(1 + Mag)
F4 = cvAdd(image_Re, image_Im, image_Re, NULL);
F5 = cvPow(image_Re, image_Re, 0.5);

A la función F1 le pasamos el ancho primero y luego el alto de la imagen y nos devuelve las dimensiones de la matriz. Después mediante la función F2 calculamos la DFT. Para calcular la magnitud (Mag = sqrt(Re^2 + Im^2)) y poder mostrarla mediante log(1 + Mag) utilizaremos las funciones F4 y F5. Después mostraremos el resultado creándonos una función para que reordene los cuadrados.

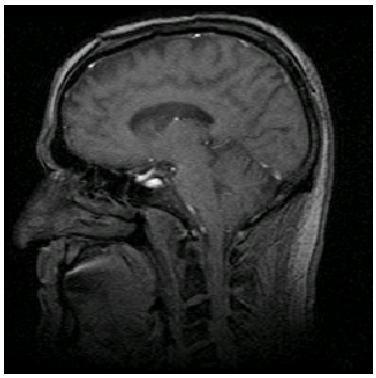
(a) Imagen girasol



(b) Transformada DFT

Una vez que nos hemos creado la matriz con los coeficientes de las distintas frecuencias de la matriz podemos seleccionar la matriz y poner los valores a cero de las frecuencias.

Mostramos un ejemplo en el que hemos puesto a cero las frecuencias de los lados izquierdo y derecho que disten del eje en vertical menos de 100.



(a) Imagen cerebro

(b) Imagen filtrada

3.3.6.- Segmentación piramidal

Los resultados en nuestra aplicación son:

(a) Imagen tigre

(b) Segmentación piramidal tigre

En la imagen b mostramos la segmentación piramidal que hemos realizado sobre la imagen a.

F1 = cvPyrSegmentation(image0, image1, storage, &comp,level, threshold1+1, threshold2+1);
La función F1 implementa el algoritmo de segmentación piramidal a esta función le debemos pasar el umbral previamente.

3.3.7.- Transformada de Hough

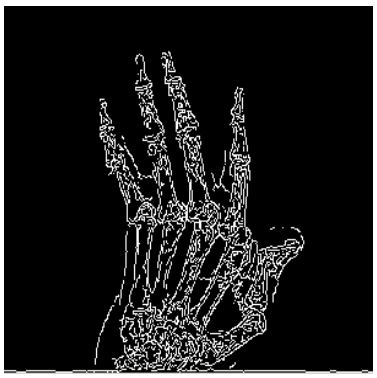
Volviendo al problema de seguimiento de bordes, la transformada de Hough se aplicaría de la siguiente manera:

Pseudocódigo del algoritmo:

Calcular el gradiente de la imagen y binarizarla con un umbral.
Especificar las divisiones del plano.
Examinar las cuentas de las celdas acumulador para altas concentraciones de píxeles.
Examinar las relaciones de continuidad entre los píxeles y una celda elegida.

El concepto de continuidad, en este caso, generalmente se está basado en el cálculo de la distancia entre píxeles desconectados identificados durante el recorrido del conjunto de píxeles en una celda acumulador dada.

(a) Radiografía de una mano



(b) Transformada de Hough

Estos resultados los hemos obtenido mediante la transformada de Hough:

F1 = cvCanny(src, dst, 50, 200, 3);
F2 = cvCvtColor(dst, color_dst, CV_GRAY2BGR);
F3 = cvHoughLines2(dst, storag, CV_HOUGH_STANDARD, 1, CV_PI/180, 100, 0, 0);

Con la función F1 aplicamos el algoritmo se Canny, después con F2 hacemos el cambio a escala de grises y por último aplicamos la transformada de Hough mediante la función F3.

3.4.- Ontologías

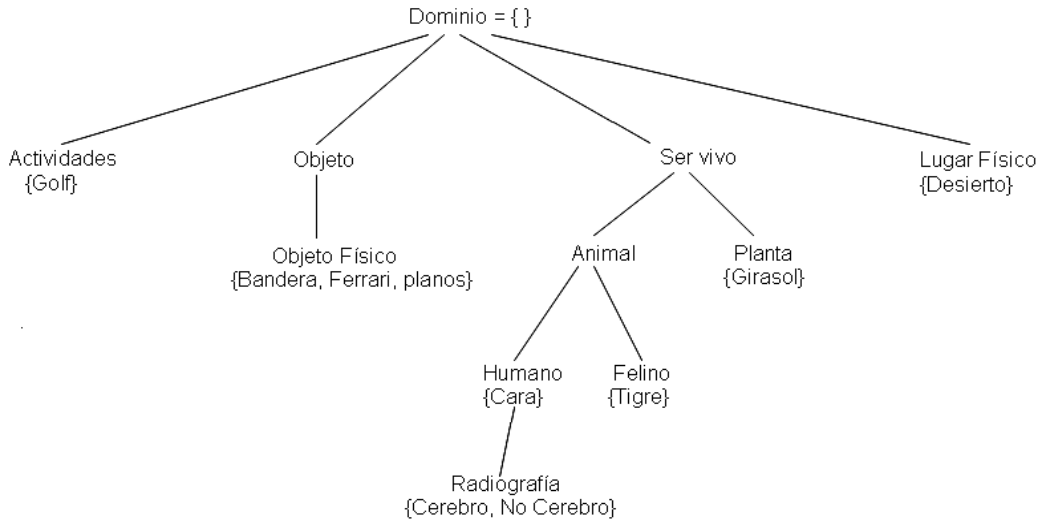
3.4.1.- Descripciones y taxonomías

El fin del proyecto era usar las ontologías para seleccionar el mejor algoritmo de segmentación.

La idea en un principio era crearse una ontología para los distintos conceptos y en función de estos conceptos y de unas propiedades de similitud entre ellos se hacia el siguiente razonamiento. Si el mejor método para el concepto C1 es el algoritmo A1, entonces si el concepto C2 esta a una distancia semántica pequeña suponemos que el algoritmo A2 también será bueno.

Ej.: El mejor algoritmo para gatos es el floodfill, tigre está cerca de gato. El mejor algoritmo para tigre será floodfill.

Para nuestro dominio la ontología sería:



Y el cuadro de texto en función de las características partonómicas, atendiendo a la clasificación de las partonomías establecida en (Winston et al, 1987) sería:

Dominio	Categoría	¿Partes ?	Parte-todo		
			1	2	3
Bandera	Objeto- Objeto físico	No			
Ferrari	Objeto- Objeto físico	Sí	X	X	X
Girasol	Ser vivo- Planta	Sí	X	X	X
Desierto	Objeto- Objeto físico	Sí	-	-	-
Golf	Actividad	Sí	X	X	X
Radio Cerebro	Ser vivo-Animal- Humano	Sí	X	X	X
Radiografía	Ser vivo-Animal- Humano	Sí	X	X	X
Plano	Objeto- Objeto físico	Sí	X	-	X
Tigre	Ser Vivo-Animal-Felino	Sí	X	X	X
Cara	Ser vivo-Animal- Humano	Sí	X	X	X

Los números 1,2,3 del cuadro son:

1) Separabilidad
2) Funcionalidad
3) Diferenciación

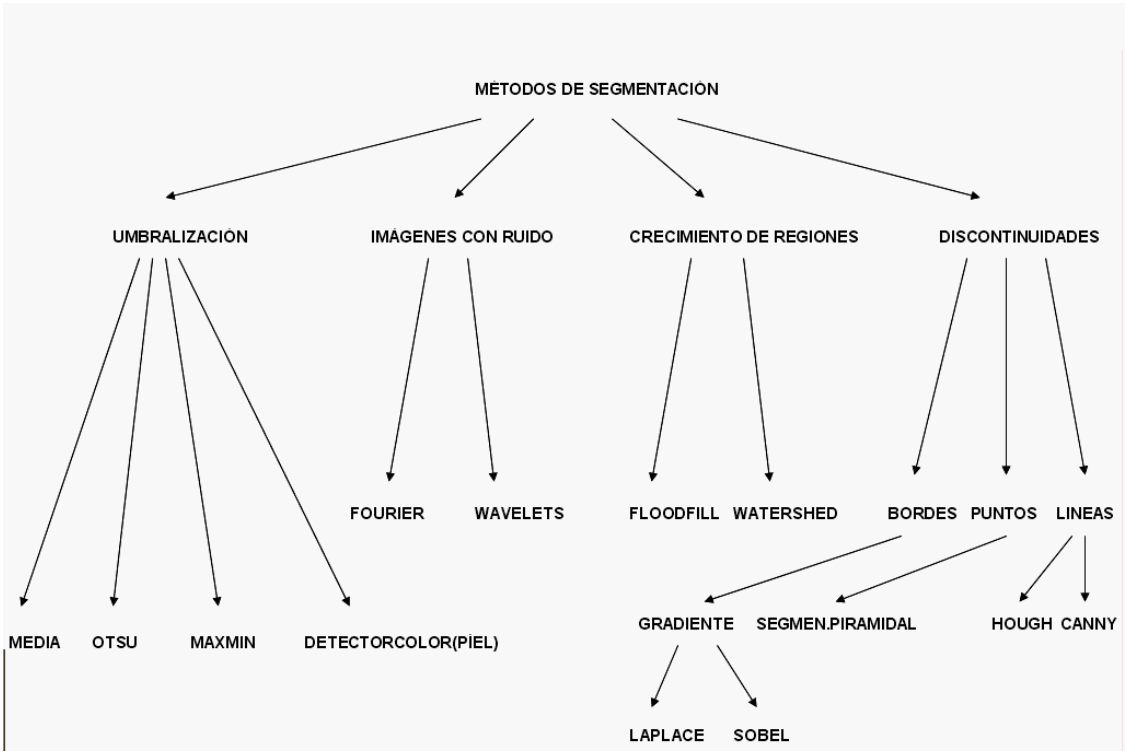
El problema para este enfoque es que el mejor algoritmo para los gatos no existe. El más adecuado puede depender de los colores del gato, del fondo en el que esté, de si la imagen es en color, de si tiene ruido, etc.

Por lo tanto si queremos seleccionar el mejor algoritmo se segmentación deberemos basarnos qué queremos extraer de la imagen, las características de la imagen y también en el caso de reconocimiento lo que esta representa [OpIn].

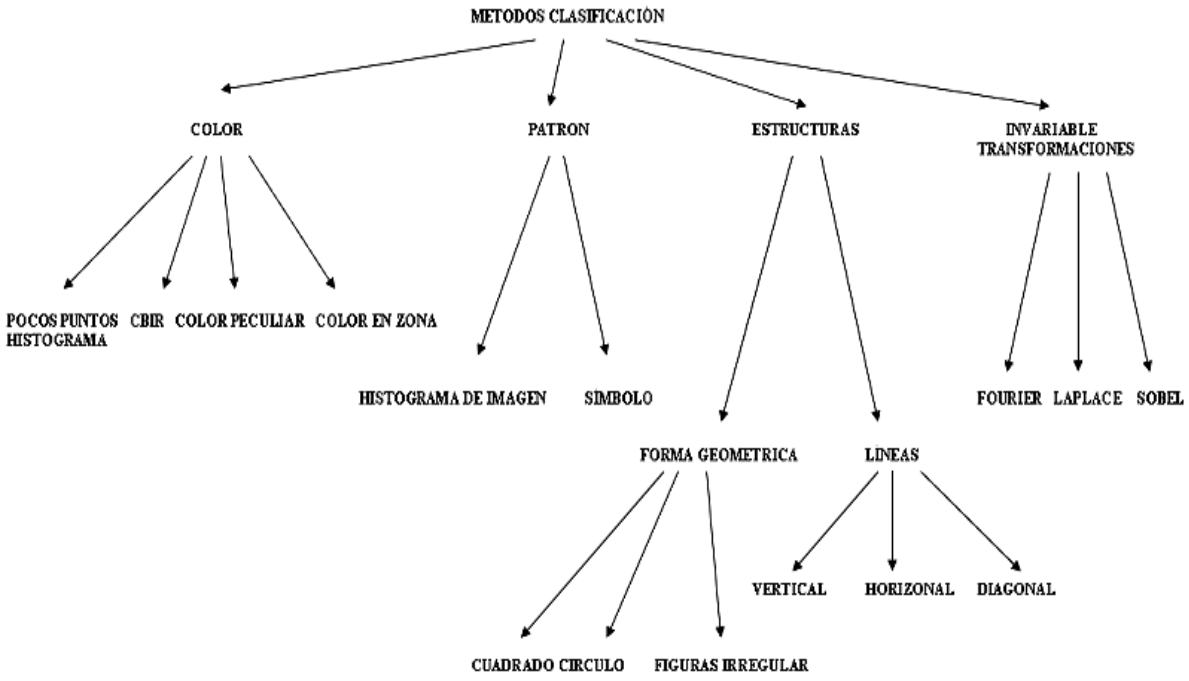
Además la segmentación no tiene un único fin. Los fines de la segmentación son: detectar objetos, detectar líneas, detectar bordes, reconocer caras, eliminar ruido, limpiar los bordes, destacar los colores, mejorar la visión, etc.

A pesar de eso, las ontologías pueden ser bastante útiles para ayudar al usuario a elegir un método de clasificación y de segmentación adecuados para la imagen que están tratando. Para ello nos crearemos una ontología basada en las propiedades de la imagen y el objetivo del usuario.

Diagrama ontología de los métodos de segmentación:



También podríamos desarrollar una ontología para los métodos de clasificación:



3.4.2.- Ontologías OWL

En este proyecto se han creado varias ontologías en el lenguaje OWL, que es el lenguaje recomendado por el W3C para el intercambio de contenidos semánticos en la Web. Hemos empleado este lenguaje para poder representar de manera compartible y reutilizable los siguientes elementos de información relacionados con este proyecto:

Características de las imágenes. Como hemos mencionado anteriormente, para cada imagen se extrae un conjunto de características, que nos servirán como base para la aplicación de la técnica adecuada de segmentación junto con la información recibida del usuario. Se ha desarrollado una ontología mínima donde se establecen las propiedades que se pueden extraer de una imagen. En nuestro

caso, esta ontología se compondrá de un único concepto (imagen), que tendrá asociada el siguiente conjunto de propiedades:

- Archivo. Ruta del archivo con la imagen
- Amarillo. Número de píxels en esta gama
- Caras. Radio del círculo que envuelve la cara
- Color. Número de valores vacíos en el histograma
- Líneas. Número de líneas que ha detectado el algoritmo

La ventaja que nos proporciona esta representación es que cualquier sistema podría extraer y procesar características de imágenes de la misma forma que nosotros empleando esta ontología, a la vez que establece un marco fácilmente extensible. Esto es debido a que la ontología podría aumentarse con más características e incluso con más conceptos sin invalidar el trabajo que ya se haya podido realizar con las imágenes a partir de la ontología actual. Este lenguaje también nos permite la definición de restricciones de cardinalidad de valores para las propiedades, por lo que es un entorno interesante para la especificación de características anotables para una imagen.

A continuación mostramos el código OWL de esta ontología, que está disponible en http://klt.inf.um.es/~jfernand/image.owl:

```
<?xml version="1.0" ?>
-<rdf:RDF
    xmlns="http://klt.inf.um.es/~lmpechuan/image.owl#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xml:base="http://klt.inf.um.es/~lmpechuan /image.owl">
  <owl:Ontology rdf:about="" />
- <owl:Class rdf:ID="IMAGE">
- <rdfs:subClassOf>
- <owl:Restriction>
- <owl:onProperty>
```

<owl:DatatypeProperty rdf:ID="color" />
</owl:onProperty>
<owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int">1</owl:cardinality>
</owl:Restriction>
</rdfs:subClassOf>
- <rdfs:subClassOf>
- <owl:Restriction>
<owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int">1</owl:cardinality>
- <owl:onProperty>
<owl:DatatypeProperty rdf:ID="lines" />
</owl:onProperty>
</owl:Restriction>
</rdfs:subClassOf>
- <rdfs:subClassOf>
- <owl:Restriction>
<owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int">1</owl:cardinality>
- <owl:onProperty>
<owl:DatatypeProperty rdf:ID="face" />
</owl:onProperty>
</owl:Restriction>
</rdfs:subClassOf>
- <rdfs:subClassOf>
- <owl:Restriction>
- <owl:onProperty>
<owl:DatatypeProperty rdf:ID="yellow" />
</owl:onProperty>
<owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int">1</owl:cardinality>
</owl:Restriction>
</rdfs:subClassOf>
- <rdfs:subClassOf>
- <owl:Restriction>
- <owl:onProperty>
<owl:DatatypeProperty rdf:ID="file" />
</owl:onProperty>
<owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int">1</owl:cardinality>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing" />
</owl:Class>
- <owl:DatatypeProperty rdf:about="#color">
<rdfs:domain rdf:resource="#IMAGE" />
<rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#float" />

```
    </owl:DatatypeProperty>
- <owl:DatatypeProperty rdf:about="#file">
  <rdfs:domain rdf:resource="#IMAGE" />
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
    </owl:DatatypeProperty>
- <owl:DatatypeProperty rdf:about="#yellow">
  <rdfs:domain rdf:resource="#IMAGE" />
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#float" />
    </owl:DatatypeProperty>
- <owl:DatatypeProperty rdf:about="#lines">
  <rdfs:domain rdf:resource="#IMAGE" />
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#float" />
    </owl:DatatypeProperty>
- <owl:DatatypeProperty rdf:about="#face">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#float" />
  <rdfs:domain rdf:resource="#IMAGE" />
    </owl:DatatypeProperty>
    </rdf:RDF>
  - <!--
   Created with Protege (with OWL Plugin 3.2.1, Build 365)  http://protege.stanford.edu
    -->
```

A continuación vemos el código OWL para la anotación de las características de imágenes particulares. Cada imagen tendrá asociada un fichero .owl que contendrá el resultado de aplicarle la extracción de características y esta descripción semántica de la imagen será empleada en posteriores tareas.

```
    <?xml version="1.0" ?>
-<rdf:RDFxmlns:image="http://klt.inf.um.es/~lmpechuan/image.owl#"
      xmlns:protege="http://protege.stanford.edu/plugins/owl/protege#"
      xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
      xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
      xmlns:owl="http://www.w3.org/2002/07/owl#"
      xmlns="http://www.owl-ontologies.com/Ontology1170754274.owl#"
      xml:base="http://www.owl-ontologies.com/Ontology1170754274.owl">
- <owl:Ontology rdf:about="">
  <owl:imports rdf:resource="http://klt.inf.um.es/~jfernand/image.owl" />
    </owl:Ontology>
- <image:IMAGE rdf:ID="foto1">
  <image:lines rdf:datatype="http://www.w3.org/2001/XMLSchema#float">25.0</image:lines>
  <image:face rdf:datatype="http://www.w3.org/2001/XMLSchema#float">55.0</image:face>
```

```
<image:file rdf:datatype="http://www.w3.org/2001/XMLSchema#string">c:\misfotos\foto1.jpg
</image:file>
 <image:color rdf:datatype="http://www.w3.org/2001/XMLSchema#float">37.0</image:color>
 <image:yellow rdf:datatype="http://www.w3.org/2001/XMLSchema#float">66.0</image:yellow>
    </image:IMAGE>
- <image:IMAGE rdf:ID="foto1">
 <image:lines rdf:datatype="http://www.w3.org/2001/XMLSchema#float">25.0</image:lines>
 <image:face rdf:datatype="http://www.w3.org/2001/XMLSchema#float">55.0</image:face>
<image:file rdf:datatype="http://www.w3.org/2001/XMLSchema#string">c:\misfotos\foto1.jpg
</image:file>
 <image:color rdf:datatype="http://www.w3.org/2001/XMLSchema#float">37.0</image:color>
 <image:yellow rdf:datatype="http://www.w3.org/2001/XMLSchema#float">66.0</image:yellow>
    </image:IMAGE>
    </rdf:RDF>
 - <!--
   Created with Protege (with OWL Plugin 3.2.1, Build 365)  http://protege.stanford.edu
   -->
```

Métodos de segmentación y de clasificación. Estas ontologías representan los tipos de segmentación y de clasificación que se comentaron anteriormente. Estas ontologías son taxonomías. Debido al propósito del proyecto, únicamente se han empleado los nombres de los nodos en la ontología creada en OWL, aunque somos conscientes de que sería adecuado definir conjuntos de propiedades distinguibles a cada nodo. Los ficheros OWL correspondientes se muestran en el anexo, y están disponibles en http://klt.inf.um.es/~jfernand/metodos.owl y http://klt.inf.um.es/~jfernand/clasificacion.owl.

3.5.- Proceso de segmentación asistida de la imagen

3.5.1.- Marco teórico

Supongamos que tenemos N dominios diferentes entre sí. Seleccionando un algoritmo de segmentación para cada dominio. Por lo tanto dada una imagen desconocida debemos de determinar a qué dominio pertenece sin la ayuda de un humano para poder seleccionar el mejor método de segmentación.

El algoritmo que tenemos posee dos fases:

Fase 1

Esta fase consiste en crearnos tuplas con distintos valores de los dominios que tenemos.

Cada tupla representa un dominio distinto y cada campo de la tupla tendrá un valor en el rango [0..1]. Este valor se lo asignaremos en función de una característica de la imagen.

Por lo tanto tenemos N dominios con M campos:

Dominio 1 = < Campo 1, Campo 2, Campo 3, Campo 4, Campo…, Campo M >
Dominio 2 = < Campo 1, Campo 2, Campo 3, Campo 4, Campo…, Campo M >
Dominio 3 = < Campo 1, Campo 2, Campo 3, Campo 4, Campo…, Campo M >
Dominio... = < Campo 1, Campo 2, Campo 3, Campo 4, Campo…, Campo M >
Dominio N = < Campo 1, Campo 2, Campo 3, Campo 4, Campo…, Campo M >

Esto lo repetiremos para todos los campos y al final tendremos una tabla con el número de Dominios y sus valores.

Quedará una tabla de la siguiente forma:

	Campo 1	**Campo 2**	**Campo 3**	**Campo 4**	**Campo …**	**Campo M**
Dominio 1	0,73192333	0,71090469	0,00627607	0,84865254	0,96772591	0,24747172
Dominio 2	0,84389551	0,63106115	0,86292451	0,99294848	0,11797304	0,82455443
Dominio 3	0,10914498	0,16042292	0,72479277	0,40108733	0,3904966	0,27839219
Dominio 4	0,12414361	0,30106611	0,98155846	0,09695028	0,28264675	0,08505404
Dominio 5	0,42570576	0,52704089	0,27986386	0,75434381	0,55818691	0,16501246
Dominio 6	0,06763507	0,7986974	0,21845716	0,67909607	0,32653475	0,10907073
Dominio 7	0,45537604	0,39498178	0,3273865	0,47414614	0,73852608	0,27596452
Dominio 8	0,45940667	0,45888254	0,89676906	0,38022242	0,65334793	0,74138293
Dominio 9	0,04424828	0,33312423	0,82437299	0,37849853	0,24785764	0,42123347
Dominio 10	0,77370893	0,14522322	0,44857161	0,63315472	0,80387754	0,09266942
Dominio …	0,80677664	0,31767281	0,07599605	0,56850199	0,98487409	0,65974793
Dominio N	0,56465946	0,59572009	0,7783062	0,95649625	0,711703	0,84651589

Fase 2

La idea es que ya tenemos una nueva imagen que no sabemos a qué dominio pertenece. A esta imagen I se le asocian los siguientes valores obtenidos por la extracción de características.

	Campo 1	**Campo 2**	**Campo 3**	**Campo 4**	**Campo …**	**Campo M**
Imagen I	0,10913796	0,05523303	0,65491346	0,11643459	0,88674816	0,96502621

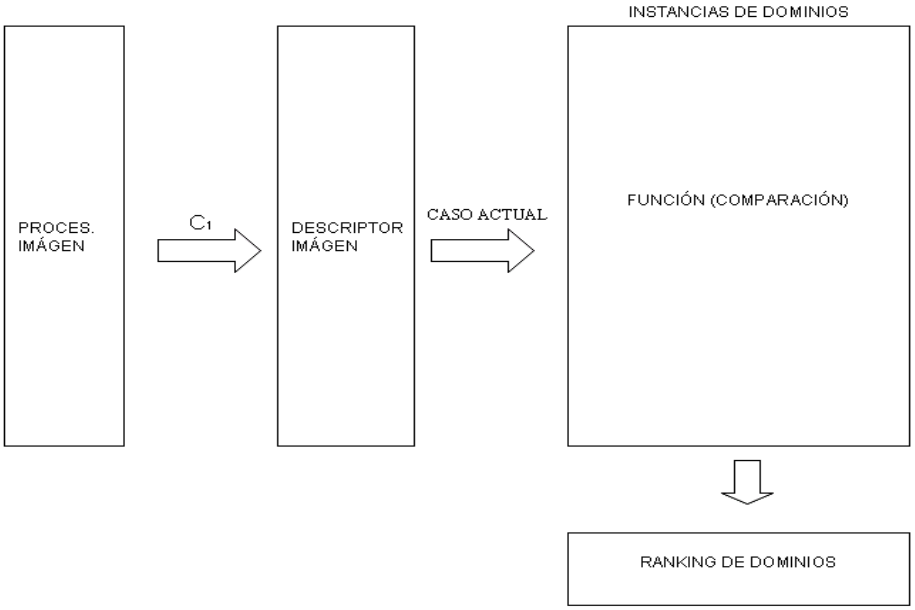
Ahora calculamos la siguiente operación para evaluar a que dominio pertenecerá:

$$\sum_{i=0}^{M} \text{ABS(Imagen.Campo(i) — Dominio.Campo(i))}$$

Esto es el sumatorio de las diferencias en valor absoluto de cada campo del dominio con la imagen.

Esta operación se realiza para todos los dominios y nos quedaremos con el dominio más pequeño, cuando no sobrepase cierto Umbral. En caso de sobrepasarlo diremos que no pertenece a ningún dominio.

Lo explicamos gráficamente:



En este gráfico vemos cómo procesamos la imagen y extraemos sus características para posteriormente creamos una tupla modulando los valores que hemos obtenido para esas características. Seleccionaremos el dominio que esté más cerca siempre que la diferencia entre los conjuntos de características esté por debajo de un umbral.

3.5.2.- Desarrollo de la aplicación

En esta aplicación tenemos tres dominios. El primer dominio que tenemos son los girasoles con los que vamos a aplicarles la segmentación piramidal. Al segundo, las caras, vamos a aplicarles el reconocimiento de caras. Al tercero, los cerebros, vamos a aplicarles el Flood Fill.

La aplicación va a consistir en dos fases.

Fase 1

En primer lugar tendremos tres dominios. Estos dominios son girasoles, cerebros y caras. Posteriormente extraeremos características a estos elementos para poder clasificarlos automáticamente.

Para los girasoles extraeremos el número de píxeles que pertenecen al amarillo del girasol.

Para el cerebro tenemos una imagen del cerebro que nos sirve de modelo y compararemos todas las imágenes con esta. En función del porcentaje que tenga de parecido con esta nueva imagen lo usaremos como otra característica de clasificación.

Para las caras le pasaremos un detector de caras que nos devolverá el radio. El radio nos servirá porque la mayoría de las imágenes de caras están a una misma distancia de la cámara. Esto es porque todas las fotos de caras son de tipo carnet.

Una vez hemos implementado los extractores de características tendremos que formar las tuplas que representen cada dominio.

Concretamente para nuestro caso es:

Dominio Cerebros = (Pixels Girasol, Parecido Cerebro, Radio Cara)
Dominio Cara = (Pixels Girasol, Parecido Cerebro, Radio Cara)
Dominio Girasol = (Pixels Girasol, Parecido Cerebro, Radio Cara)

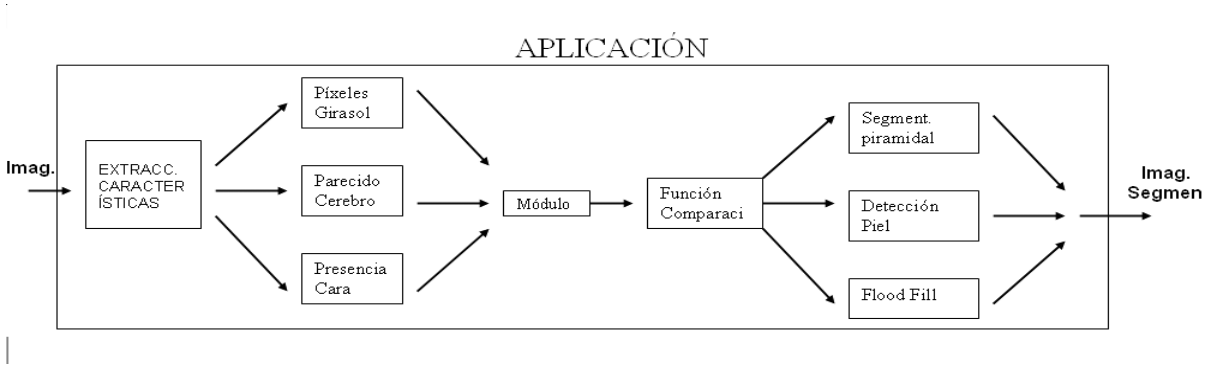
Para una tupla: Dominio = (X, Y, Z) , donde el primer valor representa el numero de pixeles que pertenecen al girasol, el segundo valor representa el parecido con la imagen de cerebro y el tercero el radio de la cara detectada. Para las 89 imágenes de cerebros el total de sus valores es (0, 7864, 841) y la media (0, 0.88, 0.00944).

Para las 49 imágenes de cara el total de sus valores es (0.0785, 2426, 3452) y la media (0.0001, 0.49, 0.70). El valor de cada indice es igual que el ane

Para las 31 imágenes de cara el total de sus valore es (149, 1453, 209) y la media (0.048, 0.46, 0.0674).

Fase 2

Mostramos un diagrama en el que tenemos una imagen como entrada, después extrae las características y modulamos de su valor. La función de comparación determina a qué dominio pertenece la nueva imagen y en función del dominio en el que esté le aplicamos una u otra técnica de segmentación.
En la figura de abajo se muestra el diagrama:



Para evaluar el rendimiento de nuestra aplicación hemos insertado 15 imágenes de caras, 12 de girasoles y 25 de cerebros. Ha clasificado correctamente todas las imágenes salvo una de caras y otra de girasoles.

3.5.3.-Inclusión de objetivos de segmentación

Como hemos comentado con anterioridad, no basta conocer el dominio para elegir un algoritmo de segmentación, sino que es necesaria información sobre qué tipo de información hay que buscar en la imagen, o qué tipo de formas existen en la misma. Combinando ambos tipos de información estaríamos en mejores condiciones de seleccionar el algoritmo de segmentación óptimo.

Basándonos en las ontologías seleccionamos unos atributos para permitir a un usuario inexperto seleccionar un algoritmo de segmentación adecuado:

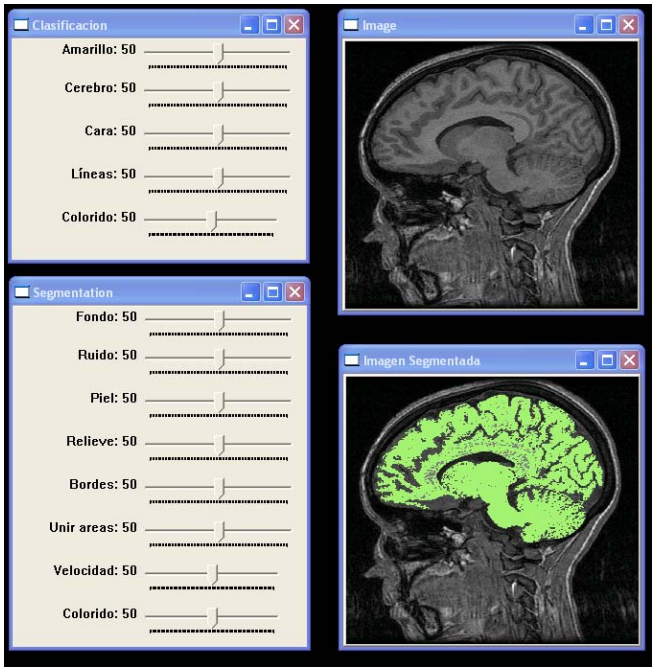
Si recordamos la ontología de los algoritmos de segmentación:

- En el caso de que el usuario quiera separar el fondo del objeto y de la velocidad con la que quiera hacerlo elegiremos entre Otsu, MinMax y Media.
- En el caso de que una imagen tenga ruido emplearemos la Transformada de Fourier.
- En el caso de que quiera agrupar por colores y del número de colores en el que quiera ver el resultado elegiremos entre Watershed, Floodfill y Segmentación piramidal.
- En el caso de marcar que quiera marcar relieves, elegiremos entre Sobel y Laplace pudiendo binarizar, en función del número de colores.
- En el caso de que quiera detectar colores de la piel usaremos un filtro detector de la piel.

- En el caso de que quiera señalar los bordes: Hough y Canny. Dependiendo del número de colores en el que quiera el resultado.

3.6.- Descripción de la aplicación

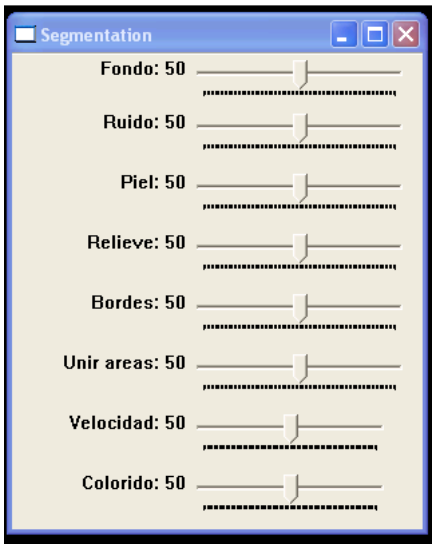
Una visión general de la aplicación sería:



Nuestra aplicación funciona de la siguiente manera:

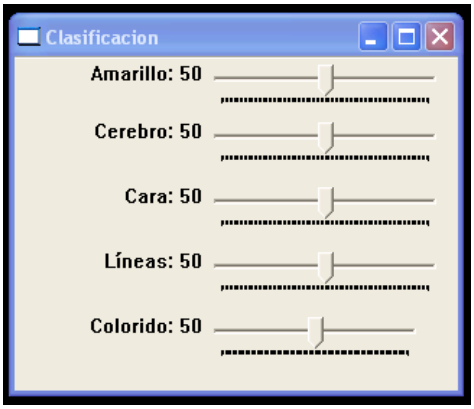
- Con el panel uno seleccionamos el tipo de imágenes que queremos segmentar.
- Con el segundo panel definimos el objetivo de la segmentación deseada.
- En la ventana Imagen, se muestra la imagen original que estamos tratando.
- En la ventana imagen segmentada se muestra el resultado de aplicarle un algoritmo de segmentación a la imagen original.

Se muestra el panel de la aplicación, donde se recogen los atributos.



Nos hemos creado una ventana con 8 parámetros. En función de los valores que elija el usuario aplicaremos un algoritmo de segmentación:

- Fondo: Este parámetro nos sirve para saber si el usuario desea separar los objetos del fondo.
- Ruido: Con este parámetro quitamos de la imagen. Las frecuencias en las que se pueda encontrar el ruido y nos permite su reconstrucción.
- Piel: Con este parámetro seleccionamos los píxeles de las imágenes que tengan tonalidades que pertenezcan a la piel.
- Relieve: Cuando el usuario quiera seleccionar algoritmos que marquen el relieve de una imagen deberá darle un valor alto a este parámetro.
- Bordes: Este parámetro lo empleamos para marcar los bordes de una imagen. Aplicaremos un algoritmo u otro en función del parámetro colorido.
- Unir áreas: Con este parámetro uniremos los colores de la imagen. Dependiendo del valor de colorido aplicaremos un algoritmo u otro.
- Velocidad: Con aplicaremos distintos tipos de algoritmos en función de la velocidad que quiera el usuario.
- Colorido: Este parámetro lo usaremos para aplicar distintos algoritmos en función del resultado que quiera el usuario.



Para la elección de imágenes hemos seleccionado 5 parámetros:

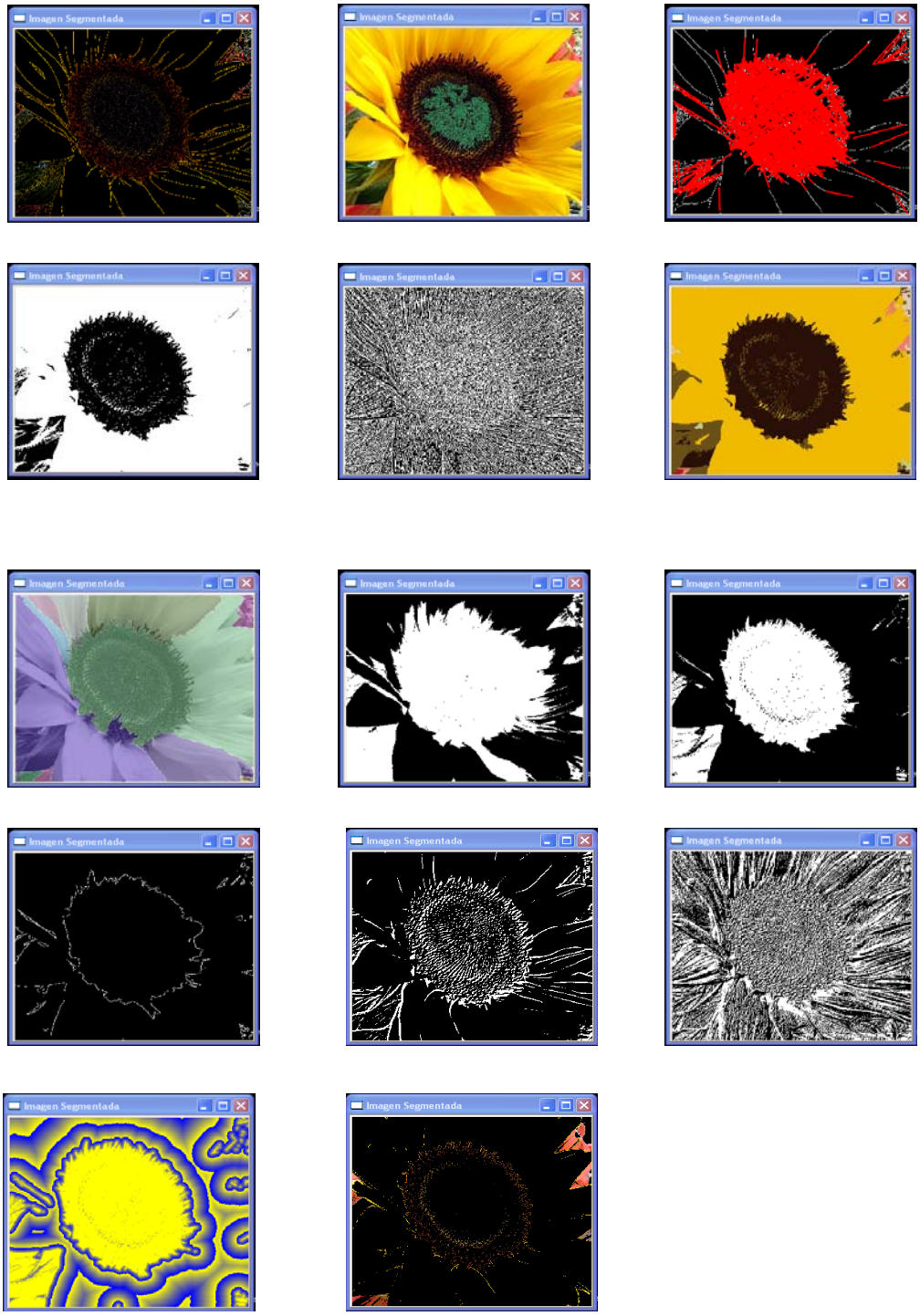
- Amarillo: Con este parámetro seleccionaremos las imágenes que tengan píxeles con la tonalidad amarilla.
- Cerebro: Con este parámetro seleccionaremos imágenes de cerebros.
- Cara: Con este parámetro seleccionamos imágenes que tengan caras.
- Líneas: Con este parámetro seleccionaremos imágenes con muchas líneas.
- Colorido: Con este parámetro seleccionamos imágenes que tengan una gran cantidad de colores.

En esta ventana de la aplicación se representa la imagen seleccionada que cumple los parámetros metidos por el usuario. En la página siguiente se muestra el resultado de aplicar los siguientes algoritmos de segmentación a esta imagen. En la siguiente figura se muestra el algoritmo empleado para cada imagen según su localización en la matriz de disposición.

a	b	c
d	e	f
g	h	i
j	k	l
m	n	

a)Canny b) FloodFill c) Hough
d) Media e) Laplace f) Segmentación piramidal
g) Watershed h) MinMax i) Otsu
j) Sobel Bin. k) Laplace Bin. l) Sobel
m) Fourier n) Detección Piel

Esta es la imagen segmentada. Según los valores que le pongamos se aplicara un algoritmo de segmentación.



CAPÍTULO 4

CONCLUSIONES Y VIAS FUTURAS

4.1.- Conclusiones

Tanto las ontologías como el tratamiento digital de imágenes son disciplinas con mucha proyección y sobre las que se han puesto muchas esperanzas. Actualmente están experimentando un gran desarrollo y son temas que suscitan un gran interés. A pesar de esto, no se encuentran muchos sistemas

donde se coopere el potencial de las ontologías con tratamiento digital de imágenes.

Las ontologías son un paso para dotar a las máquinas de la capacidad de razonar, hacer representaciones y sacar conclusiones. El tratamiento digital de imágenes es un intento de que el computador reconozca objetos en la realidad.

En este proyecto se ha hecho una simple aplicación pero nos ha servido para vislumbrar la potencia la combinación de estas dos herramientas.

Mediante Internet todos tenemos la posibilidad de acceder a documentos, sonido, video y todo tipo de información a través de nuestro ordenador pero la idea inicial que se tenia de Internet no esta del todo cumplida. Estamos en una época de transición entre Internet y Web Semántica.

La Web Semántica es una Web extendida, dotada de mayor significado en la que cualquier usuario en Internet podrá encontrar respuestas a sus preguntas de forma más rápida y sencilla gracias a una información mejor definida. Al dotar a la Web de más significado y, por lo tanto, de más semántica, se pueden obtener soluciones a problemas habituales en la búsqueda de información gracias a la utilización de una infraestructura común, mediante la cual, es posible compartir, procesar y transferir información de forma sencilla. Esta Web extendida y basada en el significado, se apoya en lenguajes universales que resuelven los problemas ocasionados por una Web carente de semántica en la que, en ocasiones, el acceso a la información se convierte en una tarea difícil y frustrante.

Por otro lado, con el tratamiento digital de imágenes podemos crear un sistema informatizado que captura la imagen de un objeto determinado y procede a la identificación de diferentes parámetros como el color la textura, la forma, la estructura interna y externa del objeto.

Estos sistemas tienen un alto nivel de flexibilidad y repetibilidad, nunca se cansan, ni se aburren ni se distraen y pueden ser puestos a trabajar en

ambientes donde los humanos no podrían trabajar bajo condiciones de seguridad.

Para un humano, los ojos proporcionan información del ambiente que lo rodea, el cerebro interpreta lo que los ojos ven basado en experiencias previas con objetos similares. Basándose en esta interpretación, se toman decisiones y acciones. En forma similar, los sistemas de visión artificial ven al objeto, lo interpretan pero necesitaríamos de las ontologias y la inteligencia artificial para razonar y extraer conclusiones.

La combinación de estas dos disciplinas daría lugar a búsquedas por lo que realmente aparece en la imagen. No como las de los búscadores de imágenes actuales que se basan únicamente en el nombre que se asocia a una imagen.

Nos damos cuenta de que al combinar estos dos elementos podemos conseguir que un robot analice una imagen, identifique los objetos y extraiga sus conclusiones.

El único problema es que en la segmentación de imágenes no se tiene una clasificación adecuada de los elementos que la forman, ni tampoco tenemos una conciencia de grupo. Da la impresión de que cada uno funciona un poco por libre pero que no hay unos objetivos comunes.

Todo lo contrario pasa en las ontologías donde es un proyecto en el que cooperan todo tipo de personas, con unos mismos objetivos y que tienen una conciencia de grupo.

4.2.- Vías futuras

4.2.1.- Selección del extractor de características

De una imagen se pueden extraerse bastantes características (la cantidad de un cierto color, el número de colores, la forma del histograma, la presencia de un patrón, etc). Pero cuando tenemos una base de datos agrupada por dominios y queremos insertar un nuevo dominio podemos desarrollarlo de distintas formas.

Pongamos el ejemplo de insertar una bombona de butano. Si queremos insertar un nuevo dominio de imágenes que contengan bombonas de butano y queremos que nuestra aplicación siga funcionando, ¿Que debemos hacer? .

La idea es que nos podemos servir de las ontologías para poder encontrar unos métodos de clasificación que sean válidos. Imaginemos que queremos insertar un nuevo dominio.

Si se lo pidiéramos a un usuario experto probablemente lo resolvería de la siguiente manera:

- Examinaría las imágenes del butano y también el resto de las imágenes.
- Buscaría algo común en todas las imágenes de butano y que ese factor común sea distinto del resto de las imágenes.
- El usuario se decantaría por el color naranja de la bombona puesto que es una característica común y excluyente del resto.
- Acceder al código de la aplicación y entenderlo.
- Debería modificarlo para que se capte esta propiedad y verificarlo.
- Compilarlo e instalarlo.

Todo este proceso se puede realizar mucho más fácilmente mediante una ontología en la que las hojas sean los métodos de selección.

Lo único que deberá hacer el usuario será responder a unas preguntas y la aplicación le dirá el método a usar, solo deberá pasarle los parámetros a ese método.
La aplicación a desarrollar haría preguntas al usuario y posteriormente analizaría sus respuestas para recomendarle un método de segmentación.

A continuación mostramos un ejemplo donde queremos insertar Planos de distintas cosas.

¿Se puede distinguir por sus colores?
De los girasoles si, de las caras también pero de los cerebros es más difícil por que también esta en blanco y negro. Es una opción pero vamos a ver si hay otra mejor.

¿Posee un patrón característico?
La verdad es que si pero tampoco esta del claro. Vamos a ver si hay otras opciones más asequibles. Aunque esta es una buena opción.

¿Permanece invariable a las transformaciones?
No lo sabemos tendríamos que aplicarlo previamente. Esta la vamos a desechar porque no parece a priori muy esperanzadora.

¿Posee estructuras?
La respuesta es que si. Los planos estas formados por líneas que representan las distintas cosas que se pueden encontrar en ellos.

¿Posee figuras geométricas?
Los planos se caracterizan por tener lineas en cualquier sentido.

¿Posee Líneas?
Si.

Podemos intentar detectar líneas verticales, horizontales y diagonales. Por lo que el mejor que puedes hacer es usar como extractor de características es la selección de líneas.

4.2.2.- Selección del método de segmentación

En esta segunda aplicación se nos presenta un nuevo problema. Vamos a intentar seleccionar el mejor algoritmo de segmentación para una imagen que no pertenece a ningún dominio. La idea es buscar el método de segmentación usando ontologías que mejor se adapte a nuestras necesidades.

Para conseguirlo habrá que hacer una ontología en la que aparezcan los métodos de segmentación con los dominios y características en las que mejor funcionan.

Para este apartado nos crearemos un diagrama con todos los algoritmos de segmentación y cada uno de ellos tiene un dominio con unas características.

Formularemos preguntas al usuario que irán recorriendo el árbol. Hasta dar con un algoritmo que se adapte a sus imágenes. Puede que se adapte o puede que sea el menos malo. Las dos cosas son muy útiles. Podemos hacer uno genérico en el caso de que no haya ninguno que case al cien por cien.

Ej.: Cuando tenemos un plano en blanco y negro, el algoritmo de segmentación que mejor funciona es el de umbralización. El problema es imaginemos un página de un libro manchada de pintura. El mejor algoritmo será el de umbralización porque son dos colores y permite separar la suciedad del texto original.

Mostramos un ejemplo:

¿Quieres eliminar el ruido?
No. Suponemos que las imágenes no están dañadas en su transmisión.

¿Quieres separarlo por regiones?
No. Lo que pretendemos es separar el blanco del negro para que se vea mejor.

¿Tiene dicontinuidades, bordes, tal o cual?
No. Ya que no hay un objeto central que queramos seleccionar. Simplemente queremos limpiar la imagen de zonas que al escasear la tiñen de grises.

¿Tiene una imagen sobre el fondo?
Si.

¿Quieres la Media?
Si.

¿Quieres la Otsu?
Si.

¿Quieres la MaxMin?
Si.

De los tres métodos que hay elegiremos el que mejor se adapte a nuestros planos.

4.2.3. – Ontología de algoritmos de segmentación

La idea de esto sería crear una página Web con algunas aplicaciones para la segmentación de imágenes. Esta aplicación debería estar dotada con todos los algoritmos de segmentación del mundo y con ciertos parámetros que introduciría el usuario.

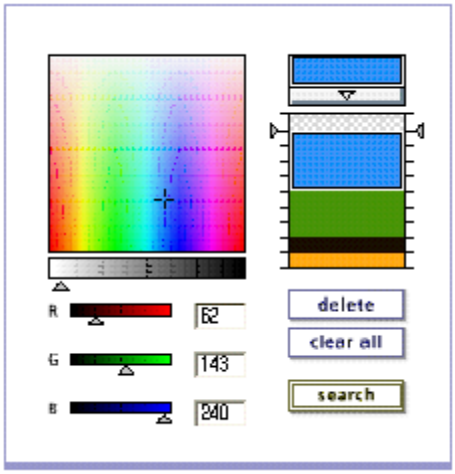
En esta aplicación se le preguntaría al usuario que es lo que quiere hacer. Es decir, mediante preguntas ir seleccionando el mejor algoritmo de segmentación para esa imagen. A veces el usuario puede que no tenga mucha idea con lo

cual no se le podrá ayudar todo lo bien que desearíamos pero siempre será una ayuda. En otro caso el usuario tendrá será más experto y le podremos hacer preguntas que requieran un nivel de conocimientos mayor.
De esa forma podrá sacar mejor rendimiento de nuestra aplicación. También se puede sugerir una lista de la cual mediante pruebas se decante por alguno.
Otra posibilidad es interaccionar con otras aplicaciones haciendo de caja negra. Es decir que nos pasen una imagen y la devolvemos segmentada, o devolvemos el algoritmo que creamos que mejor se adapta a este dominio.

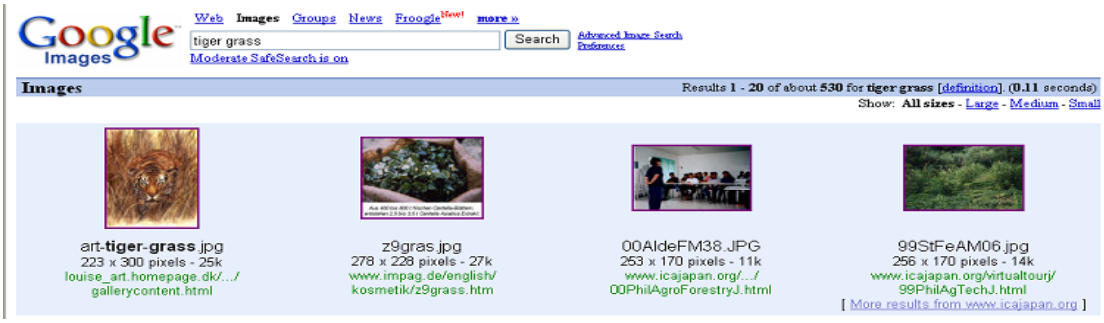
La clave aquí es juntar los conocimientos y las técnicas que se emplean en la actualidad para segmentar los tipos de imágenes. Y hacer con ellos una ontología que pueda interactuar y aconsejar a los usuarios.

4.2.4. – Buscador de imágenes

A día de hoy la mayoría de los algoritmos funciona basándose únicamente en el etiquetado de la imagen. Eso hace que no sea del todo eficiente. También existen los sistemas CBIR que se basan únicamente en el color y en las proporciones con que aparecen en la imagen pero también nos parece algo insuficiente.



Si pusiéramos en Google las palabras “Tigre, grass”, tigre y hierba en inglés, nos saldría:



Mediante las ontologías se puede obtener un mayor rendimiento. Tenemos unas imágenes que segmentamos automáticamente y extraemos los objetos. Posteriormente asociamos un nombre a cada objeto recortado. Por lo que de una imagen obtendremos varios nombres.

Para conseguir un sistema más eficiente podíamos tomar nota manualmente lo cual es poco escalable para las imágenes que hay en la red. Posteriormente podríamos automatizar este proceso.

Dada una imagen mediante distintos algoritmos, extraemos de ella los distintos tokens que son los elementos que forman la imagen. Una vez sacados los almacenamos en una base de datos relacionándola con la imagen de la siguiente manera.

Aunque esto tiene una pega y es que margen de error, la idea consiste en aplicar un filtrado a las palabras asociadas a la imagen de manera que quitemos las palabras que no tengan relación. Aquí es donde aplicamos las ontologías.

De una imagen pueden salir varios nombres. Por ejemplo; Desierto, Palmera, Piano, Serpiente. De esta forma eliminaríamos la palabra Piano porque es la que tiene menos relación con las otras. Y para ello nos vamos a basar en la distancia semántica que hay en la WordNet que es una gran ontología donde están representadas la mayoría de las palabras. Si la distancia semántica entre dos palabras es superior a un umbral en ese caso la borraremos.

REFERENCIAS

[DiIm] Digital Image Processing. Second Edition. Rafael C. Gonzalez. Richard E.Woods. Internacional Edition 2002

[TéCl] Técnicas Clásicas de Segmentación de Imagen. Marcos Martín. 4 de mayo de 2004

[Her99] Hernández, Antonio. "La Fotografía digital." Manual de documentación fotográfica. Editor Félix del Valle Gastaminza. Madrid: Síntesis, 1999. 205-229

[Jomupe] Cita 2. José Muñóz Peréz, Catedrático en Ciencias de la Computación e Inteligencia Artificial de la Universidad de Málaga.

[UmbSeg] Universidad Nacional de Quilmes – Ing. en Automatización y Control Industrial. Cátedra: Visión Artificial Octubre de 2005

[ATaPa] A Taxonomy fo Part-Whole Relations. Morton e. Winston, Roger Chaffin, Douglas Herrmann

[Acbe] Acha Piñero, Begoña. Tesis doctoral segmentación y clasificación de imágenes en color. Aplicación al diagnóstico de quemaduras. Director: José I. Acha Catalina

[DoTe] Documento de Tesis Medición automática del D-score en imágenes de muestras de endometrio humano. Javier Eduardo Rojas Romero. 9 de octubre de 2006

[ApBio] Apuntes 3 de Bioinformática. Escuela universitaria de informática. Universidad politécnica de Madrid.

[Giga] Ginés García Mateos, PROCESAMIENTO AUDIOVISUAL
Dept. de Informática y Sistemas, Universidad de Murcia

[FuVi] Fundamentos de Visión por Computador *Sistemas Informáticos Avanzados* Universidad Jaime I

[GrTr] Grupo de Trabajo en Ontologías del consorcio W3C

[UtLib] Manual: Utilización de la librería de funciones de procesamiento de imagen OpenCV .Rafael Espí Botella .Higinio Mora Mora . Francisco A. Pujol López Technical Report: I2RC-SPAL-2006-004 . Noviembre 2006.

[InMa] Manual: Open Source Computer Vision Library .Intel. *Reference Manual*

[OpIn] OpenCV : Introduction with Image & Video Processing
Yohan Chin (Jin) PhD Candidate Department of Computer Science
University of Texas Dallas

[MaJe] María Jesús Lamarca Lapuente, “Hipertexto, el nuevo concepto de documento en la cultura de la imagen” .Tesis doctoral. Universidad Complutense de Madrid http://www.hipertexto.info. 27 de diciembre 2006

[Basi] Bartolomé Sintes Marco, 6 de diciembre de 2006 http://www.mclibre.org/consultar/amaya/otros/ap_historia.html

[WiGr]William E. Green Mechanical Engineering and Mechanics Drexel University http://www.pages.drexel.edu/~weg22/can_tut.html 2002

[InTe] Instituto tecnologico de Apizaco. “Tutorial de tecnicas modernas”José Juan Hernandez Mora http://www.itapizaco.edu.mx/paginas/ttm/unid751.html

[LuGi] Luis Guillermo Restrepo Rivas Una Introducción a la Visión de Máquina 1999-2007 http://luisguillermo.com/visiona.htm

[CuVi] Curso de Visión por Computadora. Maestría en Ciencia e Ingeniería de la Computación, UNAM. Semestre 2004 http://turing.iimas.unam.mx/~elena/Teaching/CV-Mast.html

[JaKn] Jared Knutzon y Bryan Walter. “Object Recognition in Image Processing” http://www.public.iastate.edu/~knutzonj/ee424projectMain.htm 2003

[SoDe] SoftIntegration Demos for Ch and Toolkits 2006 http://www.softintegration.com/products/thirdparty/opencv/demos

[IoWa] Iowa State University http://www.hci.iastate.edu/575x/doku.php?id=krista:homework:hw4 2006

[InPr] Introduction to programming with OpenCV Gady Agam *Department of Computer Science.* Illinois Institute of Technology January 27, 2006
http://www.cs.iit.edu/~agam/cs512/lect-notes/opencv-intro/index.html

[ViSt] Visual Studio 2005 Microsoft Corporation
http://msdn2.microsoft.com/es-es/library/fx6bk1f4(vs.80).aspx 2007

[OwWe] "OWL Web Ontology Language Reference", W3C Recommendation February 2004. Disponible en: http://www.w3.org/TR/2004/REC-owl-ref-20040210/

[DepInf] Departamento de informática. Ontologías para servicios web semánticos de Información de tráfico: descripción y Herramientas de explotación. José javier samper zapater Universitat de valencia. Servei de publicacions 2005

[UsIn]Uso de Internet I: Navegando en la Web, Dr. Antonio Núñez Reis
Especialista en Medicina Intensiva. Ingeniero Informático.
Hospital Fundación Alcorcón Madrid

[ViCo] Visión por computador en sistemas de calidad Jorge Alberto Cardona Pérez, Fabio Andrés Restrepo Gómez y Diana Marcela Salazar Jiménez

ANEXO

En este apartado vamos a dar los fundamentos de las imágenes. En primer lugar haremos una Introducción. Después que diremos serán las partes del ojo. Una vez dichas daremos unas indicaciones generales de cómo funciona el sentido de las vista. Debido a que vamos a trabajar con imágenes a color daremos los fundamentos de los colores. Para concluir daremos un rápido repaso a como se forman las imágenes mediante matrices, como se almacenan en bytes, los tipos de imágenes, los dispositivos para captarlos y las distintas operaciones que podemos hacer con imágenes.

1.- Introducción

Román Gubern define la imagen como “una modalidad de la comunicación visual que representa de manera plástico-simbólica, sobre un soporte físico, un fragmento del entorno óptico (percepto), o reproduce una representación mental visualizable (ideoescena), o una combinación de ambos, y que es susceptible de conservarse en el espacio y/o en el tiempo para constituirse en experiencia vicarial óptica: es decir, en soporte de comunicación entre épocas, lugares y/o sujetos distintos, incluyendo entre estos últimos al propio autor de la representación en momentos distintos de su existencia” [Her99]. Caben pues en esta definición tanto la fotografía de un amigo como un mapa de una región como una imagen de rayos X o un cuadro impresionista.

Cuando tenemos que interpretar una imagen en muchos casos se necesitan una gran cantidad de conocimientos del tema que se trata. Podemos tomar imágenes de la sangre, la capa de ozono o del proceso de deshielo, pero para poder interpretarlas con éxito será necesario estar dotado de unos conocimientos teóricos cualificados.

En función de los dispositivos empleados para captar la realidad clasificaremos las imágenes en varios tipos.

- Imágenes formadas mediante la luz visible, es decir, la luz cuya longitud de onda oscila entre 350 y 700 nanómetros, que corresponden a las imágenes capturadas por una cámara fotográfica, de vídeo o de TV.

- Imágenes formadas por rayos infrarrojos, que permiten la visión nocturna.

- Imágenes formadas por rayos ultravioleta, como espectrogramas (fotografía de un espectro luminoso obtenida mediante un espectrógrafo, sobre placa de cristal o película sensible).

- Imágenes formadas a partir de campos magnéticos, como la resonancia magnética que utiliza un potente campo magnético para obtener imágenes detalladas del corazón o del tórax. Se coloca a la persona dentro de un gran electroimán que causa una vibración de los núcleos de los átomos del organismo, produciendo unas señales características que son convertidas en imágenes.

- Imágenes formadas a partir de ultrasonidos, como la ecografía, que forma una imagen por la reflexión de las ondas sonoras con determinadas partes del cuerpo. El ecocardiograma, que utiliza ondas ultrasonoras de alta frecuencia que chocan contra las estructuras del corazón y de los vasos sanguíneos y, al rebotar, producen una imagen móvil que aparece en una pantalla de vídeo.

- Imágenes formadas a partir de una radiación de rayos X, como las radiografías de tórax o la radioscopia que es una exploración continua con rayos X que muestra en una pantalla el movimiento del corazón y los pulmones. La tomografía computarizada que crea imágenes transversales de todo el tórax utilizando los rayos X y muestra la ubicación exacta de las anomalías (permite obtener una imagen tridimensional móvil del corazón). La angiografía muestra con detalle el aporte de sangre, por ejemplo, a los pulmones o al cerebro. Detecta anomalías vasculares como la obstrucción de un vaso sanguíneo, bolsas en una arteria o inflamaciones.

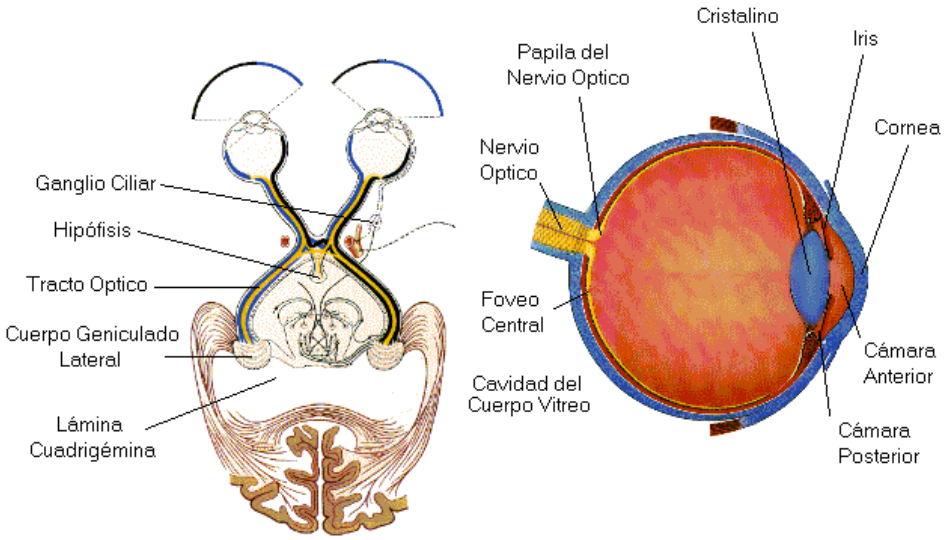
- Imágenes formadas a partir de impulsos eléctricos, como el electrocardiograma, que amplifica los impulsos eléctricos del corazón y se registran en un papel en movimiento. El electroencefalograma es una imagen formada a partir de los impulsos eléctricos de 20 alambres colocados sobre el cuero cabelludo con el objeto de establecer el trazado y registro eléctrico de la actividad cerebral.

- Imágenes formadas a partir de isótopos radiactivos. Los indicadores se reparten por todo el cuerpo y se detectan con una gammacámara (detección de coágulos de sangre en los pulmones). En la técnica de tomografía computarizada por emisión de fotones simples, distintos tipos de cámaras de

registro de radiaciones pueden grabar una imagen simple o producir una serie de imágenes de secciones transversales amplificadas por el ordenador. En las imágenes formadas por emisión de positrones, como la tomografía por emisión de positrones, se inyecta una sustancia en la sangre que se desplaza hasta las estructuras cerebrales, permitiendo medir la actividad que desarrolla el cerebro. También permite analizar, por ejemplo, el funcionamiento del corazón.

2.- Estructura del ojo

El globo ocular es una estructura esférica de aproximadamente 2,5 cm de diámetro con un marcado abombamiento sobre su superficie anterior. La parte exterior, o la cubierta, se compone de tres capas de tejido: la capa más externa o esclerótica tiene una función protectora, cubre unos cinco sextos de la superficie ocular y se prolonga en la parte anterior con la córnea transparente; la capa media o úvea tiene a su vez tres partes diferenciadas: la coroides muy vascularizada, reviste las tres quintas partes posteriores del globo ocular continúa con el cuerpo ciliar, formado por los procesos ciliares, y a continuación el iris, que se extiende por la parte frontal del ojo. La capa más interna es la retina, sensible a la luz.



La córnea es una membrana resistente, compuesta por cinco capas, a través de la cual la luz penetra en el interior del ojo. El iris es una estructura pigmentada suspendida entre la córnea y el cristalino y tiene una abertura circular en el centro, la pupila.

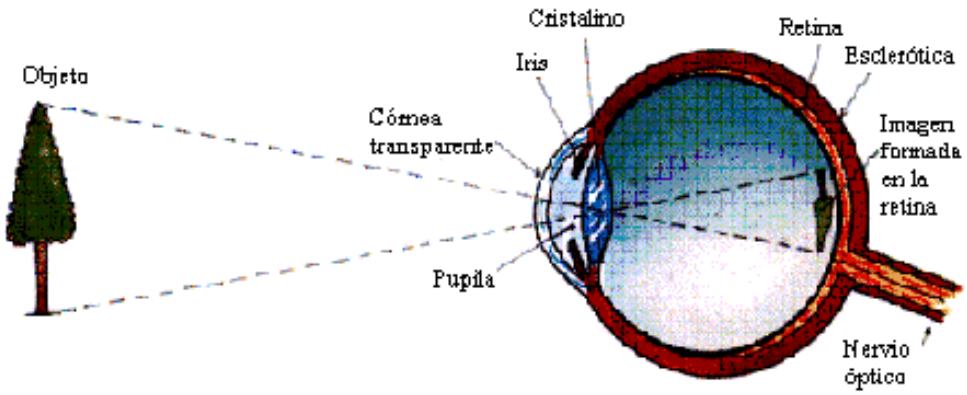
Por detrás de la lente, el cuerpo principal del ojo está lleno de una sustancia transparente y gelatinosa encerrado en un saco delgado que recibe el nombre de membrana hialoidea. La presión del humor vítreo mantiene en suspensión el globo ocular.

La retina es una capa compleja compuesta sobre todo por células nerviosas. Las células receptoras sensibles a la luz se encuentran en su superficie exterior detrás de una capa de tejido pigmentado. Estas células tienen la forma de conos y bastones y están ordenadas como los fósforos de una caja.

El nervio óptico entra en el globo ocular por debajo y algo inclinado hacia el lado interno de la fóvea central, originando en la retina una pequeña mancha redondeada llamada disco óptico.

3.- Funcionamiento del ojo

En general, las cámaras fotográficas sencillas funcionan como los ojos de los animales. La lente del cristalino forma en la retina una imagen invertida de los objetos que enfoca y la retina se corresponde con la película sensible a la luz.



El enfoque del ojo se lleva a cabo debido a que la lente del cristalino se aplana o redondea; este proceso se llama acomodación. En un ojo normal no es necesaria la acomodación para ver los objetos distantes, pues se enfocan en la retina cuando la lente está aplanada gracias al ligamento suspensorio. Para ver los objetos más cercanos, el músculo ciliar se contrae y por relajación del ligamento suspensorio, la lente se redondea de forma progresiva. Un niño puede ver con claridad a una distancia tan corta como 6,3 cm. Al aumentar la edad del individuo, las lentes se van endureciendo poco a poco y la visión cercana disminuye hasta unos límites de unos 15 cm a los 30 años y 40 cm a los 50 años. En los últimos años de vida, la mayoría de los seres humanos pierden la capacidad de acomodar sus ojos a las distancias cortas. Esta condición, llamada presbiopía, se puede corregir utilizando unas lentes convexas especiales.

4.- Colimetría

Se denomina colorimetría a la parte de la ciencia que se ocupa de la descripción y especificación adecuada del color. Dado que existen tres tipos de conos fotorreceptores, parece adecuado representar al color con tres componentes numéricas. Así se obtiene el espacio de color, que puede definirse como el conjunto de todos los colores, estando cada uno de ellos especificado mediante un vector tridimensional. Según la definición de cada una de las componentes se derivarán distintos espacios de color [Acbe].

En el tratamiento del color existen unos parámetros básicos cuyas definiciones se presentan a continuación:

a) Intensidad: Es una medida, sobre una parte del espectro electromagnético, del flujo de potencia que es radiado desde una superficie, o incidente en ella, y se expresa en unidades de vatios por metro cuadrado.

b) Brillo: Es un atributo relacionado con la sensación visual de que un área parezca que emita más o menos luz.

c) Luminancia: Es la potencia radiada ponderada por una función de sensibilidad espectral característica de la visión humana. Fue definida por la CIE (siglas francesas de la Comisión Internacional de Iluminación).

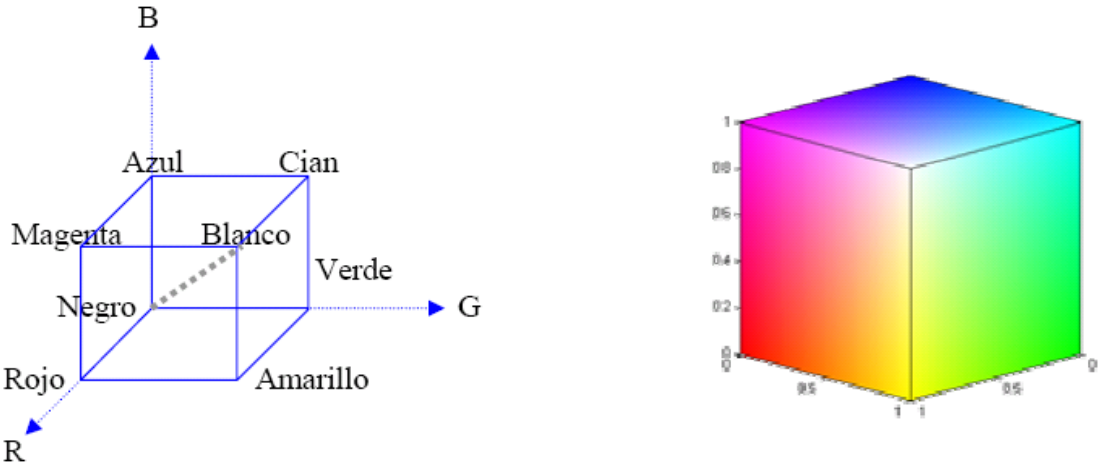
d) Luminosidad: Es la respuesta perceptiva no lineal a la luminancia que tiene la visión humana. La no linealidad es prácticamente logarítmica.

e) Tono: Es una propiedad del color asociada con la longitud de onda dominante en una mezcla de ondas de luz, por tanto, representa el color dominante tal y como es percibido por un observador. Así, cuando se dice que un objeto es rojo, verde, amarillo o de otro color, es su tono lo que se especifica.

f) Saturación: Se refiere a la pureza relativa o a la cantidad de luz blanca mezclada con un tono. Los colores puros del espectro están completamente saturados y no contienen luz blanca. El grado de saturación disminuye a medida que se añade más luz blanca.

El tono y la saturación describen la crominancia. Los humanos interpretan el color basándose en su luminosidad, tono y saturación.

El color es una propiedad de gran importancia en la percepción visual humana. El color está asociado con la capacidad de los objetos de reflejar ondas electromagnéticas de diferente longitud de onda. El ser humano detecta los colores como combinaciones de tres colores primarios, el rojo, el verde y el azul. Las pantallas de nuestros ordenadores suelen utilizar el modelo BGR, donde cada píxel tiene asociado un vector tridimensional (B,G,R) que nos conduce a un color determinado; el vector (0,0,0) es el negro, (255,255,255) es el blanco, (0,0, 255) es el rojo “puro”, y así sucesivamente. Esto implica que tenemos 3 colores diferentes que no todos serán albergados por el dispositivo, por lo que es común especificar un subconjunto de estos, llamado paleta de colores.



Espacio de colores BGR.

Los colores rojo, verde y azul, que corresponde a vértices de cubo son los colores primarios de la luz y los colores cian, magenta y amarillo de los otros tres vértices son los colores secundarios.

5. - Imagen digital

Cualquier imagen digital se puede modelarse por una función continua de dos o tres variables. En el caso de imágenes estáticas los dos argumentos de la función son las coordenadas (x,y) del plano cartesiano, mientras que si las imágenes van cambiando en el tiempo necesitamos una tercera coordenada t que nos especifica el tiempo. Los valores de la función corresponden a la

$$\text{punto de la imagen} \equiv (x, y, t) \xrightarrow{f} f(x, y, t) \equiv \text{intensidad luminosa}$$

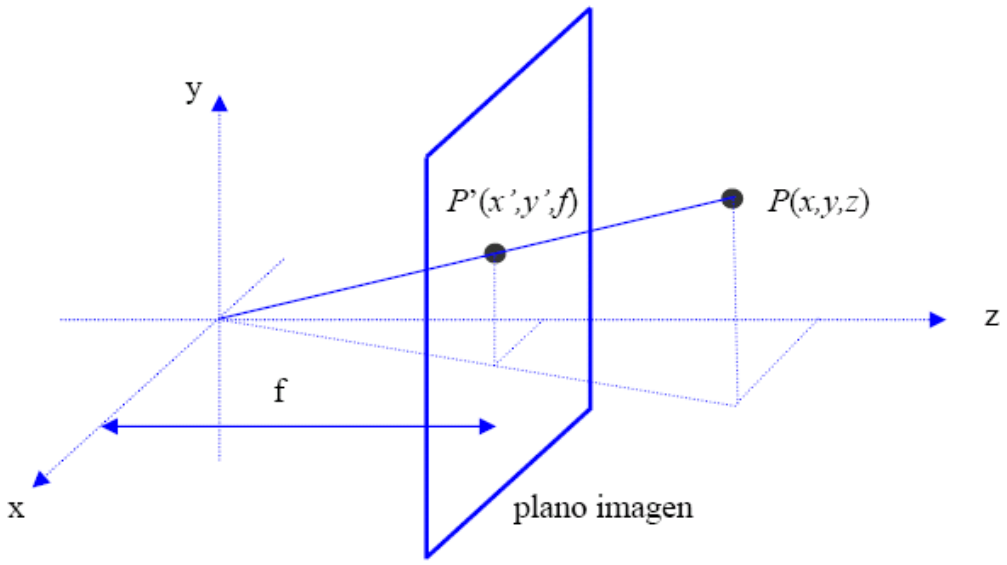
luminosidad, brillo o intensidad de la radiación de los puntos de la imagen. También pueden representar temperatura, presión, distancia al observador, etc.

Una función continua tiene el dominio y el rango continuo; una función discreta tiene el dominio discreto y la función digital tiene tanto el dominio como el rango discretos.

Una imagen 2D es el resultado de una proyección de una escena 3D. Un punto de la escena 3D viene representado por su proyección en el plano imagen. Así, el punto P=(x,y,z) se proyecta en el punto P'=(x',y',f) cuyas coordenadas vienen dadas por las expresiones:

$$x' = \frac{xf}{z}, \qquad y' = \frac{yf}{z}$$

que se obtienen directamente del teorema de Thales.



Un píxel de una imagen digitalizada es un punto del dominio de la función digital correspondiente que tiene asociado el valor de dicha función y cuya posición viene determinada por sus coordenadas (x, y). En el análisis de imágenes juegan un papel importante los píxeles de un entorno (píxel próximos) de cada píxel. Por ello, es necesario definir una función distancia entre píxeles.

La distancia Euclídea entre los píxeles (i, j) y (h, k) viene dada por la expresión:

$$D_E[(i,j),(h,k)] = \sqrt{(h-i)^2 + (k-j)^2}$$

En una imagen monocromática (en blanco y negro o en tonos de gris) el rango de f tiene un valor mínimo (negro) y un valor máximo (blanco). Si se trata de una imagen digital el rango viene dado por un conjunto finito de valores, como puede ser el conjunto {0, 1, ..., L-1, L}, donde el 0 corresponde al negro y el valor L a blanco. En este caso diremos que la imagen tiene L+1 niveles o tonos de gris. El dominio de la función digital es también una región acotada del plano, D, de la forma:

$$D = \{ (x, y): x = 1,2,...,M, y = 1,2,...,N \}.$$

Por lo tanto, una imagen digital monocromática viene dada por una matriz f de tamaño M×N:

$$f = \begin{bmatrix} f(1,1) & f(1,2) & \dots & f(1,N) \\ f(2,1) & f(2,2) & \dots & f(2,N) \\ \dots & \dots & \dots & \dots \\ f(M,1) & f(M,2) & \dots & f(M,N) \end{bmatrix}$$

Cada uno de sus elementos se llama píxel y f(m, n) nos da el tono de gris del píxel que ocupa la posición (m, n).

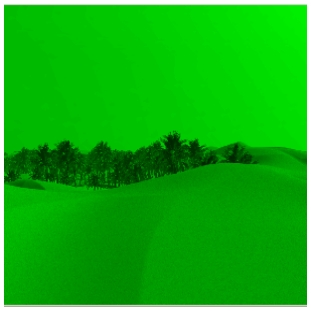
Una imagen de color RGB viene definida por tres matrices de tamaño M×N, de manera que cada píxel corresponde a una posición específica en las mismas y tiene asociado una tripleta de valores, (r, g, b), que indican la tonalidad corresponde a cada uno de los colores rojo, verde y azul. La primera matriz nos da la componente roja, la segunda la componente verde y la tercera la componente azul.

En la figura del desierto se muestran los tres canales:

(a) Imagen en color.

(b) Componente azul.

(c) Componente roja.



(d) Componente verde.

La resolución espacial viene dada por la proximidad de la muestras de la imagen en el plano imagen. En el caso de imágenes digitalizadas es el número de puntos del dominio de la imagen, es decir, M × N. La resolución espectral viene dada por el ancho de banda de las frecuencias de la radiación luminosa capturadas por el sensor. La resolución radiométrica viene dada por el número de niveles o tonos de gris distinguibles y la resolución temporal viene dada por el intervalo de tiempo entre muestras consecutivas en las que la imagen se captura.

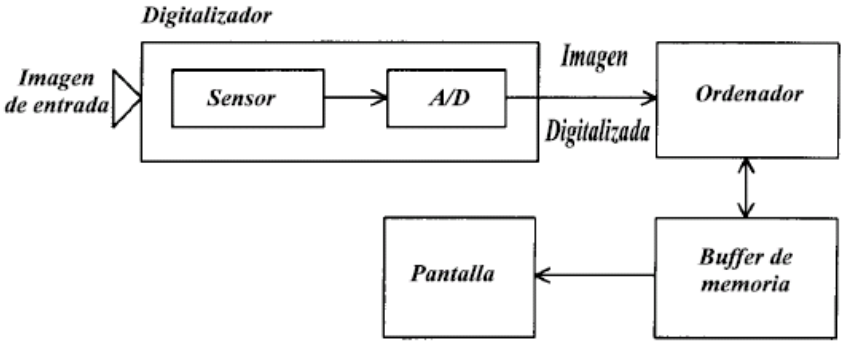
6.- Adquisición de imágenes

A la adquisición de imágenes le llamamos rasterización, que es el proceso por el cual una imagen real es captada mediante un dispositivo y transformada en una imagen digital para su visualización y procesamiento en un ordenador.

El procesamiento de imagen llevado a cabo por ordenador, se denomina procesamiento de imagen digital y se lleva a cabo, normalmente, sobre imágenes reales previamente rasterizadas. Las aplicaciones de procesamiento de imagen digital son muy variadas, siendo utilizado en:

- Investigaciones Biológicas: para el análisis de muestras biológicas.
- Defensa: Reconocimiento de formas, mejora e interpretación de imágenes.
- Imágenes de diagnóstico médico.
- Publicidad
- Fotografía
- Exploración del espacio y astronomía.
- Efectos especiales en vídeos y películas.

Antes de que un ordenador pueda comenzar a procesar una imagen, este imagen debe estar primero disponible en formato digital. Esto se lleva a cabo mediante dispositivos conocidos como Sistemas de adquisición de una imagen digital (SAID), que es un dispositivo capaz de escanear una escena y generar un vector de números representando las intensidades de luz en un conjunto discreto de puntos.



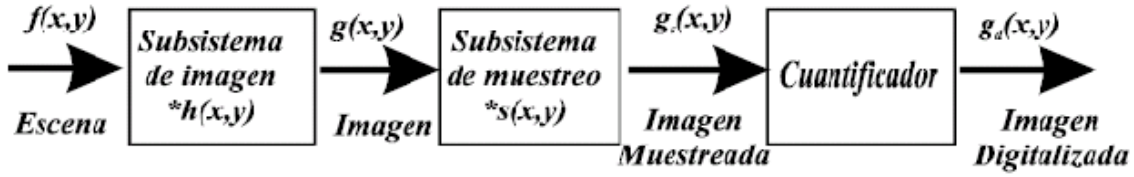
Un sistema de adquisición de imagen digital consta de tres componentes básicos: un sensor de imagen que mide la cantidad de luz, un escáner que recoge las medidas a través de toda la escena y un convertidor analógico-digital que discretiza los valores continuos en números de precisión finita adecuados para el procesamiento por ordenador.

El proceso completo de digitalización de una imagen puede ser descompuesto en las siguientes fases:

- Captación de imagen: mediante un dispositivo hardware sensible a la intensidad de luz externa.

- Muestreo de imagen: que transforma la imagen de entrada en una imagen espacialmente discreta.

- Cuantificación de la imagen: que discretiza los valores de intensidad de entrada en función de la precisión del ordenador. Las cuantificaciones pueden ser uniformes o no uniformes respecto a las intensidades de entrada.



La digitalización de imágenes es, pues, el resultado de un muestreo espacial y una cuantificación de intensidades. A cada elemento de muestreo se le denomina Píxel. Cualquier sistema de procesamiento digital de imagen toma como entrada una matriz bidimensional de píxeles. Cada píxel se puede ver como un área finita rectangular de la pantalla del ordenador.

7- Tipos de imágenes

Imágenes generadas por ordenador de forma interactiva:

a) Imágenes de arte: creadas normalmente sobre una base interactiva entre el hombre y una máquina. Las imágenes pueden ser reales, abstractas o simbólicas pero el motivo de su creación es de expresión artística[Jomupe].

b) Imágenes de diseño:
i) Gráficos estadísticos: que representan gráficamente series de datos numéricos con el fin de mejorar la comunicación y la comprensión en la interpretación de datos.
ii) Gráficos de noticias: que representan visualmente normalmente un hecho o una situación geográfica para ayudar a contextualizar una noticia (dibujos de las partes de un barco, dibujos que representan la secuencia de hechos en un atraco a un banco, etc.). El fin es mejorar la comunicación de cierto tipo de información.
iii) Gráficos de mapas: que representan un lugar geográfico.
iv) Gráficos de diseño industrial: que ayudan al diseño de edificios, planos de componentes eléctricos, de patronaje, etc.
v) Gráficos de entretenimiento: dibujos de personajes u objetos para juegos, etc.

Imágenes generadas por medios distintos al ordenador y capturadas luego a través de escáner o cámara digital para ser tratadas por un ordenador.

a) Imágenes de documentos impresos: cuyo contenido más importante es texto, aunque también pueden contener gráficos. Son normalmente documentos impresos que se capturan normalmente a través de un escaner.

b) Imágenes fotográficas: captadas de la realidad directamente por cámaras fotográficas digitales o por satélites o por escaners a partir de fotos convencionales, bien procedentes de papel, de negativos o de diapositivas.

c) Imágenes de mediciones: son representaciones de medidas, por ejemplo de colores, para detectar bancos de pesca o distintas temperaturas o ultrasonogramas y escáneres para identificar diferencias en la densidad de los cuerpos.

8.- Principales tipos de procesamientos de imágenes

8.1.- Operaciones individuales

Las operaciones individuales implican la generación de una nueva imagen modificando el valor de cada píxel de acuerdo con una regla global aplicada a cada localización de la imagen original [ApBio].

El nuevo valor de nivel de gris de cada píxel se obtiene de acuerdo con q(x,y) = f(p(x,y)) donde la función f puede ser un operador lineal o no lineal.

- Operador inverso:

$$q = 255 - p$$

- Operador umbral binario:

Esta transformación crea una imagen de salida binaria a partir de una imagen en escala de grises; el nivel de transición está dado por el parámetro de entrada p_1.

$$q = \begin{cases} 0 \text{ si } p \leq p_1 \\ 255 \text{ si } p > p_1 \end{cases}$$

- Operador umbral en escala de grises:

$$q = \begin{cases} 0 \text{ si } p \leq p_1 \text{ o si } p \geq p_2 \\ p \text{ si } p_1 < p < p_2 \end{cases}$$

- Operador adición:

Utiliza la información contenida en la misma localización de dos imágenes distintas, A y B, para crear una nueva imagen C. La dimensión de las imágenes ha de ser la misma.

$$c(x,y) = (a(x,y) + b(x,y))/k \text{ con } k=2,3$$

para dos, tres, imágenes de entrada. Suele utilizarse para reducir el efecto del ruido.

- Operador sustracción:

$$c(x,y) = k(a(x,y) - b(x,y))$$

donde k es una función no lineal para que el valor mínimo de c(x,y) sea 0 y el máximo 255 y no haya valores negativos. Se utiliza para detectar movimientos.

- Transformaciones lógicas:

Con imágenes binarizadas es posible realizar operaciones lógicas como negación, or, and… y todo tipo de operaciones relacionales.

- Transformaciones geométricas:

El interés principal de realizar una operación geométrica sobre una imagen es observarla desde otro punto de vista. Para simular una rotación, por ejemplo, los parámetros necesarios son el ángulo de giro y las coordenadas del punto sobre el que se gira la imagen.

8.2.- Operaciones con conjuntos de píxeles

Dada una imagen f(i,j) se obtiene a partir de ella una nueva imagen g(i,j) promediando para cada píxel (i,j) los valores de intensidad de los píxeles incluidos en un entorno de vecindad de (i,j), previamente definido.

- Filtro de la mediana:
Es un filtro de orden. Opera en la vecindad de un determinado píxel, ordenándolos por valores crecientes de nivel de gris y reemplazándolo por el valor central (la mediana).

- Filtro de la media aritmética:

$$g(x,y) = \frac{1}{nm} \sum f(x,y)$$

donde nm es el nº de píxeles de la ventana. Este filtro suaviza las variaciones locales.

- Ecualización del histograma:
Esta técnica realza el contraste de la imagen original mediante una expansión de los niveles de gris de la misma. Si N(f) es el nº de píxeles del nivel f de grises y NM es el nº total de píxeles de la imagen, entonces la probabilidad para cada nivel de gris es

$$P(f) = \frac{N(f)}{NM} \qquad f=0,1,2,\ldots,255$$

y $\Sigma\, p(f)=1.$

Y la ecualización uniforme se obtiene con $G(f)=255\Sigma p(f)$

- Operadores de primera derivada:

La derivada primera es cero en las zonas de intensidad constante de una imagen y varía en las zonas de cambio de intensidad.El gradiente de una imagen f(x,y) en un punto (x,y) se define como un vector de dos componentes

G[f(x,y)]= [Gx , Gy] con Gx= ∂'x(f(x,y) y Gy= ∂'y(f(x,y)
Habitualmente,|G| ≈ | Gx | +| Gy |

En la práctica hay diferentes maneras de calcular la derivada de la imagen, dando lugar a diferentes operadores.

- Operadores de Sobel:
Las máscaras utilizadas para obtener las componentes del gradiente son:

$$\begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix} \text{ para } G_x \text{ y } \begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix} \text{ para } G_y$$

Cada píxel de la imagen original se sustituye por el producto de sus vecinos por las máscaras correspondientes y de ahí:

Gx = (z3+2z6 +z9)-(z1+2z4+z7) y
Gy = (z7+2z8+z9)-(z1+2z2+z3)

- Operador de Previtt:
Se diferencia del operador de Sobel en el valor de los coeficientes:

$$\begin{pmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{pmatrix} \text{ para } G_x \text{ y } \begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{pmatrix} \text{ para } G_y$$

- Operador de Roberts:

Operador más simple que los anteriores, donde las componentes del vector derivada se obtienen como el valor absoluto de la diferencia en niveles de grises, entre los vecinos diagonales. Así, cada píxel (x,y) se sustituye por

$$|f(x,y)-f(x-1,y-1)|+|f(x,y-1)-f(x-1,y)|$$

- Operadores de segunda derivada:

La segunda derivada es cero en todos los puntos de la imagen excepto al principio y al final de una transición de intensidad. Luego un cambio de intensidad se manifiesta como un cambio de signo de esta segunda derivada.

- Operador Laplaciana:

La laplaciana de una función f(x,y) es un vector que tiene por componentes las derivadas parciales segundas de la función. En forma discreta se puede implementar de diferentes modos, por ejemplo

$$\nabla 2\ f = 4z5 -(z2+z4+z6+z8)$$

utilizando como máscaras

$$\begin{pmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{pmatrix} \text{o} \begin{pmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{pmatrix}$$

8.3.- Filtrado en frecuencias

Las imágenes digitales se pueden interpretar en función de sus frecuencias espaciales obtenidas mediante la transformada rápida de Fourier. En este dominio de frecuencias, las imágenes también pueden ser manipuladas mediante máscaras que permiten interceptar determinadas frecuencias de la imagen. Una vez “filtradas” en el dominio de las frecuencias, se deshace el cambio obteniendo la nueva imagen.

- Filtrado paso bajo:

Los bordes y otras transiciones bruscas en los niveles de grises, como el ruido, están asociados a las frecuencias altas de la transformada de Fourier (TF), de una imagen. Por lo tanto, suprimiendo un rango de frecuencias altas, se consigue un suavizado de la imagen.

Tenemos G(u,v)=H(u,v)F(u,v) donde F(u,v) es la TF de la imagen f(x,y) y H(u,v) es la máscara que debemos aplicar para obtener G(u,v), cuya antitransformada nos dará la imagen g(x,y) deseada con D0 positivo.

$$H(u,v) = \begin{cases} 1 \text{ si } D(u,v) \le D_0 \\ 0 \text{ si } D(u,v) > D_0 \end{cases}$$

- Filtrado paso alto:

$$H(u,v) = \begin{cases} 0 \text{ si } D(u,v) \le D_0 \\ 1 \text{ si } D(u,v) > D_0 \end{cases}$$

- Filtrado paso banda:

$$H(u,v) = \begin{cases} 0 \text{ si } D(u,v) \le D_1 \text{ o si } D(u,v) \ge D_2 \\ 1 \text{ si } D_1 < D(u,v) < D_2 \end{cases}$$

9.- Dominios de trabajo

Los 10 dominios de los que hemos hablado son:

Golf

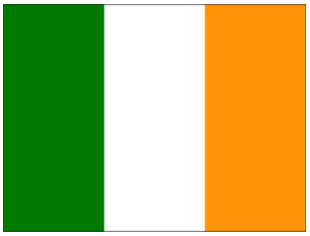



Caras

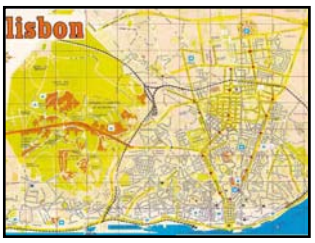


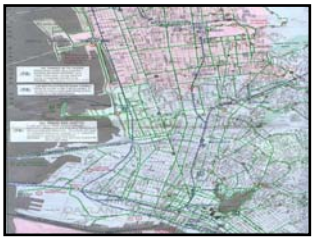

Banderas






Planos



Girasoles

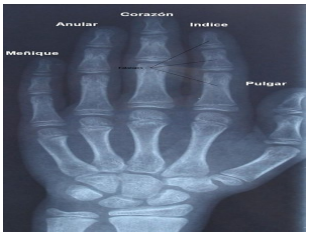



Desierto

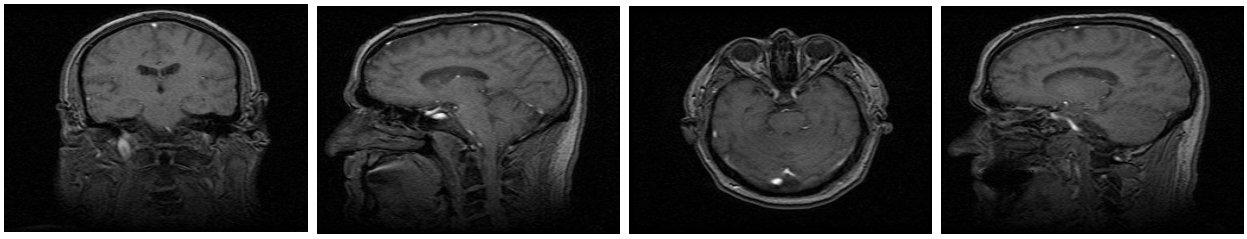



Radiografías





Radiografías cerebro



Tigres

Ferrari

10. Ontología OWL de métodos de segmentación

```
<?xml version="1.0"?>
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns="http://klt.inf.um.es/~jfernand/metodos.owl#"
  xml:base="http://klt.inf.um.es/~jfernand/metodos.owl">
  <owl:Ontology rdf:about=""/>
  <owl:Class rdf:ID="LAPLACE">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="GRADIENTE"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="SOBEL">
    <rdfs:subClassOf>
      <owl:Class rdf:about="#GRADIENTE"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="UMBRALIZACION">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="METODO_DE_SEGMENTACION"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="PUNTOS">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="DISCONTINUIDADES"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="MAXMIN">
    <rdfs:subClassOf rdf:resource="#UMBRALIZACION"/>
  </owl:Class>
  <owl:Class rdf:ID="SEGMENTACION_PIRAMIDAL">
    <rdfs:subClassOf rdf:resource="#PUNTOS"/>
  </owl:Class>
  <owl:Class rdf:ID="DETECTOR_COLOR_PIEL">
    <rdfs:subClassOf rdf:resource="#UMBRALIZACION"/>
  </owl:Class>
  <owl:Class rdf:ID="WATERSHED">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="CRECIMIENTO_DE_REGIONES"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="MEDIA">
    <rdfs:subClassOf rdf:resource="#UMBRALIZACION"/>
  </owl:Class>
  <owl:Class rdf:ID="CANNY">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="LINEAS"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="OTSU">
    <rdfs:subClassOf rdf:resource="#UMBRALIZACION"/>
```

```
  </owl:Class>
  <owl:Class rdf:about="#CRECIMIENTO_DE_REGIONES">
    <rdfs:subClassOf rdf:resource="#METODO_DE_SEGMENTACION"/>
  </owl:Class>
  <owl:Class rdf:ID="FOURIER">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="IMAGENES_CON_RUIDO"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:about="#LINEAS">
    <rdfs:subClassOf>
      <owl:Class rdf:about="#DISCONTINUIDADES"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="BORDES">
    <rdfs:subClassOf>
      <owl:Class rdf:about="#DISCONTINUIDADES"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="FLOODFILL">
    <rdfs:subClassOf rdf:resource="#CRECIMIENTO_DE_REGIONES"/>
  </owl:Class>
  <owl:Class rdf:about="#IMAGENES_CON_RUIDO">
    <rdfs:subClassOf rdf:resource="#METODO_DE_SEGMENTACION"/>
  </owl:Class>
  <owl:Class rdf:ID="HOUGH">
    <rdfs:subClassOf rdf:resource="#LINEAS"/>
  </owl:Class>
  <owl:Class rdf:about="#GRADIENTE">
    <rdfs:subClassOf rdf:resource="#BORDES"/>
  </owl:Class>
  <owl:Class rdf:about="#DISCONTINUIDADES">
    <rdfs:subClassOf rdf:resource="#METODO_DE_SEGMENTACION"/>
  </owl:Class>
  <owl:Class rdf:ID="WAVELET">
    <rdfs:subClassOf rdf:resource="#IMAGENES_CON_RUIDO"/>
  </owl:Class>
</rdf:RDF>

<!-- Created with Protege (with OWL Plugin 3.2.1, Build 365)
http://protege.stanford.edu -->
```

11. Ontología OWL de métodos de clasificación

```
<?xml version="1.0"?>
<rdf:RDF
    xmlns="http://klt.inf.um.es/~jfernand/clasificacion.owl#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
  xml:base="http://klt.inf.um.es/~jfernand/clasificacion.owl">
  <owl:Ontology rdf:about=""/>
  <owl:Class rdf:ID="ESTRUCTURAS">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="METODOS_DE_CLASIFICACION"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="FORMA_GEOMETRICA">
    <rdfs:subClassOf rdf:resource="#ESTRUCTURAS"/>
  </owl:Class>
  <owl:Class rdf:ID="PATRON">
    <rdfs:subClassOf rdf:resource="#METODOS_DE_CLASIFICACION"/>
  </owl:Class>
  <owl:Class rdf:ID="CIRCULO">
    <rdfs:subClassOf rdf:resource="#FORMA_GEOMETRICA"/>
  </owl:Class>
  <owl:Class rdf:ID="POCOS_PUNTOS_HISTOGRAMA">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="COLOR"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="SIMBOLO">
    <rdfs:subClassOf rdf:resource="#PATRON"/>
  </owl:Class>
  <owl:Class rdf:ID="DIAGONAL">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="LINEAS"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="CBIR">
    <rdfs:subClassOf>
      <owl:Class rdf:about="#COLOR"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="INVARIABLE_TRANSFORMACIONES">
    <rdfs:subClassOf rdf:resource="#METODOS_DE_CLASIFICACION"/>
  </owl:Class>
  <owl:Class rdf:ID="CUADRADO">
    <rdfs:subClassOf rdf:resource="#FORMA_GEOMETRICA"/>
  </owl:Class>
  <owl:Class rdf:ID="VERTICAL">
    <rdfs:subClassOf>
      <owl:Class rdf:about="#LINEAS"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:about="#COLOR">
    <rdfs:subClassOf rdf:resource="#METODOS_DE_CLASIFICACION"/>
  </owl:Class>
```

```
  <owl:Class rdf:ID="FOURIER">
    <rdfs:subClassOf rdf:resource="#INVARIABLE_TRANSFORMACIONES"/>
  </owl:Class>
  <owl:Class rdf:ID="SOBEL">
    <rdfs:subClassOf rdf:resource="#INVARIABLE_TRANSFORMACIONES"/>
  </owl:Class>
  <owl:Class rdf:ID="HISTOGRAMA_DE_IMAGEN">
    <rdfs:subClassOf rdf:resource="#PATRON"/>
  </owl:Class>
  <owl:Class rdf:ID="HORIZONTAL">
    <rdfs:subClassOf>
      <owl:Class rdf:about="#LINEAS"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="LAPLACE">
    <rdfs:subClassOf rdf:resource="#INVARIABLE_TRANSFORMACIONES"/>
  </owl:Class>
  <owl:Class rdf:about="#LINEAS">
    <rdfs:subClassOf rdf:resource="#ESTRUCTURAS"/>
  </owl:Class>
  <owl:Class rdf:ID="COLOR_PECULIAR">
    <rdfs:subClassOf rdf:resource="#COLOR"/>
  </owl:Class>
  <owl:Class rdf:ID="COLOR_EN_ZONA">
    <rdfs:subClassOf rdf:resource="#COLOR"/>
  </owl:Class>
</rdf:RDF>

<!-- Created with Protege (with OWL Plugin 3.2.1, Build 365)
http://protege.stanford.edu -->
```