



UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA  
Escuela de Ingeniería Informática



Trabajo Fin de Grado

# Pezqueñines: Aplicación para dispositivos móviles de clasificación de la talla de peces

**Autora:** Natalia de Jesús Peñate Quintana

**Tutores:** Modesto F. Castrillón Santana (Dpto. Informática y Sistemas)

Javier Lorenzo Navarro (Dpto. Informática y Sistemas)

Antonio José Sánchez López (Singular Games)

# Agradecimientos

Dar las gracias en primer lugar a mis padres por animarme y apoyarme a la hora de volver a las aulas, en un momento de mi vida en el más lo necesitaba.

A Gustavo Medina por darme la oportunidad de realizar las prácticas en su empresa, y proponerme este proyecto, así como asignarme al tutor Antonio José Sánchez, el cual que ha estado en todo momento respaldándome y ayudándome en su realización.

Gracias a mis compañeros de prácticas y de empresa, por estar ahí ofreciéndome su apoyo y colaboración, en especial a Adae por sus aportaciones en el desarrollo Android.

Quisiera hacer extensiva mi gratitud a mis tutores en la universidad, Modesto F. Castrillón y Javier Lorenzo, por el tiempo empleado en la supervisión, seguimiento y orientación en todo el proceso de desarrollo.

Finalmente a todas esas personas que han creído en mí, familiares y amigos, por su comprensión y su motivación, especialmente a Marina Peñate por su apoyo incondicional.

# ÍNDICE

<b>1. ESTADO ACTUAL Y OBJETIVOS .....</b>	<b>4</b>
1.1 OBJETIVOS .....	6
<b>2. COMPETENCIAS QUE HAY QUE CUBRIR.....</b>	<b>8</b>
<b>3. APORTACIONES.....</b>	<b>10</b>
<b>4. NORMATIVA Y LEGISLACIÓN.....</b>	<b>11</b>
4.1 NORMATIVA Y REGULACIÓN DE LA INFORMÁTICA EN EL ÁMBITO INTERNACIONAL. ....	11
4.2 NORMATIVA Y REGULACIÓN DE LA INFORMÁTICA EN EL ÁMBITO EUROPEO. ....	14
4.3 NORMATIVA Y REGULACIÓN DE LA INFORMÁTICA EN EL ÁMBITO NACIONAL .....	15
4.3.1 <i>Leyes y Decretos Ley</i> .....	15
4.3.2 <i>Código Penal</i> .....	16
4.3.3 <i>Recomendaciones de la APD</i> .....	17
4.4 LA PROTECCIÓN JURÍDICA DE PROGRAMAS DE ORDENADOR. PIRATERÍA INFORMÁTICA.....	18
4.4.1 <i>¿En qué casos se infringe la Ley?</i> .....	19
4.4.2 <i>Medidas Judiciales</i> .....	20
4.5 NORMATIVA Y REGULACIÓN DE TALLAS MÍNIMAS .....	20
4.5.1 <i>Legislación Comunitaria</i> .....	22
4.5.2. <i>Legislación Nacional</i> .....	23
4.6 GNU GENERAL PUBLIC LICENSE.....	23
4.7. SOFTWARE LIBRE .....	24
4.8. LICENCIA DE SOFTWARE .....	24
<b>5. DESARROLLO .....</b>	<b>26</b>
5.1 METODOLOGÍA .....	26
5.2 ETAPAS DEL TRABAJO .....	27
5.2.1 <i>Análisis de requisitos</i> .....	28
5.2.1.1 Esquema de Viola-Jones para la detección de objetos.....	28
5.2.1.2 OpenCV.....	32
5.2.1.3 Comparativa entre plataformas móviles. ....	37
5.2.1.4 Plataforma de trabajo: Android .....	39
5.2.1.5 Eclipse.....	43
5.2.1.6 OpenCV en Android.....	44
5.2.1.7 Requisitos de hardware y software. ....	45
5.2.1.8 Casos de uso.....	46
5.2.2 <i>Diseño</i> .....	48
5.2.2.1 Interfaz gráfica. ....	48
5.2.2.2. Diagrama de flujo.....	49
5.2.3 <i>Implementación</i> .....	50
5.2.3.1 Entrenamiento de los detectores. ....	50
5.2.3.2. Desarrollo del software prototipo de detección. ....	60
<b>6. CONCLUSIONES .....</b>	<b>65</b>
6.1 TRABAJOS FUTUROS .....	67
<b>7. BIBLIOGRAFÍA .....</b>	<b>69</b>
<b>ANEXO 1: INSTALACIÓN DEL ENTORNO DE DESARROLLO ANDROID.....</b>	<b>71</b>
<b>ANEXO 2: GUÍA DE INSTALACIÓN DE OPENCV PARA WINDOWS .....</b>	<b>76</b>

# 1. ESTADO ACTUAL Y OBJETIVOS

España ha sido y es una de las grandes potencias pesqueras mundiales. Así lo indican el tamaño de su flota (tonelaje y potencia), el volumen de capturas y el valor de la pesca desembarcada.

Nuestro país tiene un amplio perímetro costero cuyo litoral se reparte entre mares diferentes: el Océano Atlántico y el Mar Mediterráneo.

El Océano Atlántico, con las diferencias lógicas entre latitudes tiene unas aguas de salinidad moderada, corrientes marinas que facilitan la distribución del plancton y una oscilación del nivel de las aguas de hasta cuatro metros por efecto de las mareas. Todo ello permite la existencia de una franja costera de varios hectómetros de anchura, (sumergida y emergida), que facilita el marisqueo sobre la arena de la playa.

El Mediterráneo es un mar de aguas calientes. Contiene menos fitoplancton que el océano, y la salinidad se eleva. La comunicación con el Atlántico es escasa, por lo que el Mediterráneo es frágil a efectos ecológicos. Las diferencias marinas y litorales justifican la diversidad de la fauna y su proverbial riqueza, destaca la sardina, la merluza, el atún, el mero, etc.

Las diferencias entre uno y otro mar explican el distinto significado de la pesca en cada una de las regiones costeras, las cuales nuestros mares han perdido importancia pesquera en los últimos años debido a la sobreexplotación.

España pertenece actualmente a la Europa comunitaria, por lo que participa de la Política Pesquera Común (PPC) que guarda gran afinidad con la Política Agrícola Común (PAC) conteniendo cuatro puntos básicos:

1. **Política de conservación de recursos:** Con este fin se establecen las Tarifas Anuales de Capturas, base para la asignación de cuotas a los países miembros.

2. **Política estructural:** Orientada a la mejora de las estructuras pesqueras, de la industria transformadora y de los equipamientos portuarios, trata de ajustar la flota a las disponibilidades de pesca.

3. **Organización Común de Mercados (OCM):** Tiene una gran similitud con la agraria y está encaminada a establecer y garantizar rentas equitativas a los pescadores, y precios razonables a los consumidores.

4. **Política de acceso a los caladeros exteriores:** Se realiza mediante acuerdos de diversa naturaleza con terceros países para que los buques de los países comunitarios puedan pescar en sus respectivos caladeros.

Como consecuencia de la entrada de España en la Unión Europea, el Fondo de Regulación y Organización del Mercado de los Productos de la Pesca y Cultivos Marinos (FROM) [1], organismo encargado de la promoción del consumo de productos pesqueros, creado en 1980 [2], es desde 1986 hasta 1991 el único Organismo de la Administración Pesquera española con competencias en la OCM de productos de la pesca.

Una de las primeras campañas de FROM con mayor repercusión tuvo lugar en 1983, con un llamativo anuncio de televisión, usado hasta 1993, en el que peces de colores de dibujos animados, caracterizados como bebés con grandes chupetes, interpretaban una alegre canción. Su letra aconsejaba no consumir peces inmaduros, y utilizaba el eslogan «Pezqueñines ¡No gracias!».



*Figura 1.1- Logotipo de la campaña «Pezqueñines ¡No gracias!»*

Con esta campaña se consiguió que toda España se sensibilizara en el no consumo de peces inmaduros. Se considera que un pez es inmaduro cuando el ejemplar no supera la talla mínima [7] al ser capturado, no ha madurado y, por lo tanto, no ha tenido la oportunidad de reproducirse. Como consecuencia, se limita la regeneración del recurso y la pervivencia de las especies.

Es necesario proteger nuestros escasos recursos, evitar el consumo de peces inmaduros, y con la irrupción actual de teléfonos inteligentes (smartphones) equipados con diversos sensores, se abre un amplio abanico de posibilidades para el desarrollo de aplicaciones que contribuyan a ello. En este trabajo se estudia la posibilidad que este tipo de dispositivos tiene para llevar a cabo el control de talla en pesquerías, dado que hasta el momento solo existen aplicativos en los que el usuario puede consultar dichas tallas, como por ejemplo en Pescar en Asturias Lite [19] o en bóVEDA [20], pero ninguna de ellas realiza dicha comprobación de forma interactiva analizando el tamaño del pez, únicamente ofrecen esa información.

En 2012 desaparece el FROM como organismo autónomo en el nuevo organigrama del Ministerio de Agricultura, Alimentación y Medio Ambiente. Su actividad quedará adscrita a la Secretaría General de Agricultura y Alimentación. En concreto, serán dos los departamentos que recibirán las funciones anteriormente concentradas en el FROM: el FEGA (Fondo Español de Garantía Agraria), que asumirá la tramitación de los fondos comunitarios (FEMP); y la Dirección General de la Industria Alimentaria.

## **1.1 OBJETIVOS**

El objetivo principal de este proyecto es el desarrollo de una aplicación para dispositivos móviles, que permita determinar si un pez alcanza la talla mínima para el consumo.

Para conseguir dicho objetivo debemos desarrollar utilidades básicas para implementar un prototipo para plataformas móviles, que realice la detección y segmentación de un pez, para posteriormente determinar si cumple con la talla mínima establecida para su consumo. Esta comprobación la realizaremos utilizando como referencia una moneda de un euro para calibrar el tamaño. Asimismo necesitaremos

haber realizado una serie de detectores, que nos permita la detección tanto del pez como de la moneda.

Durante el desarrollo del proyecto nos fijaremos algunos hitos que desarrollaremos en diferentes etapas. Los hitos principales son los siguientes:

**1. Estudiar y comprender el algoritmo de Viola-Jones y la librería OpenCV.**

En primer lugar nos centraremos en comprender cómo funciona el algoritmo de detección de objetos de Viola-Jones [3] [4], lo que supondrá parte de los conocimientos previos que necesitaremos para desarrollar el proyecto. Y también estudiaremos cómo funciona la librería OpenCV [5] que nos permitirá utilizarlo en un entorno real.

**2. Entrenar clasificadores.** Una vez adquiridos los conocimientos necesarios realizaremos un entrenamiento [6], con el trabajo previo de recopilación de muestras positivas y negativas, de manera que obtengamos los detectores necesarios para las especies de peces seleccionadas en el prototipo, y la moneda.

**3. Estudiar las alternativas para el desarrollo de aplicaciones móviles.**

Existen muchas plataformas para móviles (iPhone, Symbian, Windows Phone, BlackBerry, Palm, Java Mobile Edition, Linux Mobile (LiMo),...); sin embargo presentan una serie de características diferentes, las cuales analizaremos para seleccionar, la más adecuada a nuestro proyecto.

**4. Realizar una aplicación prototipo para Smartphone.**

Por último, una vez terminadas las etapas anteriores y seleccionada la plataforma móvil más adecuada, desarrollaremos sobre la misma un prototipo aplicativo que cumpla el objetivo principal de nuestro proyecto.

## **2. COMPETENCIAS QUE HAY QUE CUBRIR**

### **CII01**

*Capacidad para diseñar, desarrollar, seleccionar y evaluar aplicaciones y sistemas informáticos, asegurando su fiabilidad, seguridad y calidad, conforme a principios éticos y a la legislación y normativa vigente.*

Esta competencia queda cubierta con lo planteado en el capítulo de “Desarrollo”. En él se muestra como ha sido todo el proceso de desarrollo del proyecto con el fin de crear un producto que satisfaga la demanda, asegurando así la fiabilidad, seguridad y calidad del mismo. Además de poner de manifiesto los conocimientos adquiridos durante la realización de los estudios, aplicándolos correctamente para el buen desarrollo de este trabajo de fin de grado.

### **CII02**

*Capacidad para planificar, concebir, desplegar y dirigir proyectos, servicios y sistemas informáticos en todos los ámbitos, liderando su puesta en marcha y su mejora continua y valorando su impacto económico y social.*

Dicha competencia la podemos ver el apartado de “Aportaciones”, así como en el de “Desarrollo” en el que se explica las fases del proyecto y la metodología que se va a utilizar.

### **CII04**

*Capacidad para elaborar el pliego de condiciones técnicas de una instalación informática que cumpla los estándares y normativas vigentes.*



Para el desarrollo e instalación de este trabajo se han tenido en cuenta los requisitos hardware y software fundamentales, estos se encuentran documentados en el capítulo “Desarrollo” de la presente memoria.

## **CII18**

*Conocimiento de la normativa y la regulación de la informática en los ámbitos nacional, europeo e internacional.*

Esta competencia queda cubierta en el capítulo “Normativa y legislación”, detallando la normativa y la regulación de la informática relacionada en los ámbitos nacional, europeo e internacional y en el ámbito del propio trabajo de fin de grado.

## **TFG01**

*Ejercicio original a realizar individualmente y presentar y defender ante un tribunal universitario, consiste en un proyecto en el ámbito de las tecnologías específicas de la Ingeniería en Informática de naturaleza profesional en el que se sinteticen e integren las competencias adquiridas en las enseñanzas.*

Dicha competencia queda cubierta con la realización de este trabajo, la memoria del mismo y su posterior defensa ante el tribunal.

### **3. APORTACIONES**

La irrupción actual de smartphones equipados con diversos sensores, abre las posibilidades a numerosas aplicaciones. Por otro lado, la extracción de recursos naturales como son las pesquerías se deben hacer de forma respetuosa y permitiendo la regeneración de los mismos para evitar la sobreexplotación. Con este trabajo pretendemos aprovechar las posibilidades que este tipo de dispositivos para llevar a cabo el control de talla en pesquerías.

Este proyecto desarrolla un medio para estimar la medida de especies marinas, con el cual se pueda posteriormente determinar si un ejemplar de dicha especie cumple la talla mínima establecida para su consumo, y así colaborar en la protección de peces inmaduros, fomentando el desarrollo sostenible de nuestros mares.

Este proyecto puede ser utilizado en diferentes ámbitos o utilizarlo de manera diferente en muchas aplicaciones. En relación con la pesca marítima de recreo, una de las actividades de recreo con más tradición y con más practicantes en el litoral español, su continuidad depende de la sostenibilidad de los recursos pesqueros, por lo que los usuarios y quienes se despliegan económicamente a su servicio están interesados en su preservación. Este proyecto podría ser utilizado por los mismos en el desarrollo de su actividad, contribuyendo a una pesca sostenible.

Asimismo podría ser utilizado por el Servicio de Protección de la Naturaleza (SEPRONA), encargado del cumplimiento de las normas relativas a caza, pesca y conservación de bosques.

Por otro lado este trabajo sirve de ayuda para el desarrollo de aplicativos en los se pretenda la detección de cualquier objetos, o como en este caso, en la detección simultánea de varios objetos diferentes.

## **4. NORMATIVA Y LEGISLACIÓN**

### ***4.1 Normativa y regulación de la informática en el ámbito internacional.***

En el contexto internacional, son pocos los países que cuentan con una legislación apropiada. Entre ellos, se destacan Alemania, Argentina, Austria, Chile, España, Estados Unidos, Francia, Gran Bretaña y Holanda.

Dado lo anterior, a continuación se mencionan algunos aspectos relacionados con la ley en los diferentes países, así como con los delitos informáticos que persigue.

#### **Alemania**

Este país sancionó en 1986 la Ley contra la Criminalidad Económica, que contempla los siguientes delitos:

- Espionaje de datos.
- Estafa informática.
- Alteración de datos.
- Sabotaje informático.

#### **Austria**

La Ley de reforma del Código Penal, sancionada el 22 de Diciembre de 1987, sanciona a aquellos que con dolo causen un perjuicio patrimonial a un tercero influyendo en el resultado de una elaboración de datos automática a través de la confección del programa, por la introducción, cancelación o alteración de datos o por actuar sobre el curso del procesamiento de datos. Además, contempla sanciones para quienes comenten este hecho utilizando su profesión de especialistas en sistemas.

## **Chile**

Chile fue el primer país latinoamericano en sancionar una Ley contra delitos informáticos, la cual entró en vigencia el 7 de junio de 1993. Esta ley se refiere a los siguientes delitos:

- La destrucción o inutilización de los datos contenidos dentro de una computadora es castigada con penas de prisión. Asimismo, dentro de esas consideraciones se encuentran los virus.
- Conducta maliciosa tendiente a la destrucción o inutilización de un sistema de tratamiento de información o de sus partes componentes o que dicha conducta impida, obstaculice o modifique su funcionamiento.
- Conducta maliciosa que altere, dañe o destruya los datos contenidos en un sistema de tratamiento de información.

## **Estados Unidos**

Este país adoptó en 1994 el Acta Federal de Abuso Computacional, que modificó al Acta Federal de Fraude y Abuso Computacional de 1986.

Con la finalidad de eliminar los argumentos hipertécnicos acerca de qué es y qué no es un virus, un gusano, un caballo de Troya y en qué difieren de los virus, la nueva acta proscribire la transmisión de un programa, información, códigos o comandos que causan daños a la computadora, a los sistemas informáticos, a las redes, información, datos o programas. La nueva ley es un adelanto porque está directamente en contra de los actos de transmisión de virus.

Asimismo, en materia de estafas electrónicas, defraudaciones y otros actos dolosos relacionados con los dispositivos de acceso a sistemas informáticos, la legislación estadounidense sanciona con pena de prisión y multa a la persona que defraude a otro mediante la utilización de una computadora o red informática.

En el mes de julio del año 2000, el Senado y la Cámara de Representantes de este país, tras un año largo de deliberaciones, establece el Acta de Firmas Electrónicas en el Comercio Global y Nacional. La ley sobre la firma digital responde a la necesidad de dar validez a documentos informáticos, mensajes electrónicos y contratos establecidos mediante Internet, entre empresas (para el B2B) y entre empresas y consumidores (para el B2C).

## **Francia**

En enero de 1988, este país dictó la Ley relativa al fraude informático, en la que se consideran aspectos como:

- Intromisión fraudulenta que suprima o modifique datos.
- Conducta intencional en la violación de derechos a terceros que haya impedido o alterado el funcionamiento de un sistema de procesamiento automatizado de datos.
- Conducta intencional en la violación de derechos a terceros, en forma directa o indirecta, en la introducción de datos en un sistema de procesamiento automatizado o la supresión o modificación de los datos que éste contiene, o sus modos de procesamiento o de transmisión.
- Supresión o modificación de datos contenidos en el sistema, o bien en la alteración del funcionamiento del sistema (sabotaje).

## **Gran Bretaña**

Debido a un caso de hacking en 1991, comenzó a regir en este país la Computer Misuse Act (Ley de Abusos Informáticos). Mediante esta ley, el intento, exitoso o no, de alterar datos informáticos es penado con hasta cinco años de prisión o multas. Esta ley tiene un apartado que especifica la modificación de datos sin autorización.

## **Holanda**

El 1 de Marzo de 1993 entró en vigencia la Ley de Delitos Informáticos, en la cual se penaliza los siguientes delitos:

- El hacking y el preacking (utilización de servicios de telecomunicaciones evitando el pago total o parcial de dicho servicio).
- La ingeniería social (arte de convencer a la gente de entregar información que en circunstancias normales no entregaría).
- La distribución de virus.

## ***4.2 Normativa y regulación de la informática en el ámbito europeo.***

Hasta ahora, el principal esfuerzo europeo por regular el tema de los delitos informáticos dio como resultado el Convenio sobre la Ciberdelincuencia, de 21 de noviembre de 2001. Este documento fue firmado por los representantes de cada país miembro del Consejo de Europa, aunque su eficacia depende de su posterior refrendo por los órganos nacionales de cada país firmante.

El Convenio sobre la Ciberdelincuencia permitió la definición de los delitos informáticos y algunos elementos relacionados con éstos, tales como sistemas informáticos, datos informáticos, o proveedor de servicios. Estos delitos informáticos fueron clasificados en cuatro grupos:

1. Delitos contra la confidencialidad, la integridad y la disponibilidad de los datos y sistemas informáticos.
  - Acceso ilícito a sistemas informáticos.
  - Interceptación ilícita de datos informáticos.
  - Interferencia en el sistema mediante la introducción, transmisión, provocación de daños, borrado, alteración o supresión de éstos.
  - Abuso de dispositivos que faciliten la comisión de delitos.
2. Delitos informáticos.
  - Falsificación informática que produzca la alteración, borrado o supresión de datos informáticos que ocasionen datos no auténticos.
  - Fraudes informáticos.

3. Delitos relacionados con el contenido.
  - Delitos relacionados con la pornografía infantil.
4. Delitos relacionados con infracciones de la propiedad intelectual y derechos afines.

Conviene destacar que en el Convenio sobre la Ciberdelincuencia se encomienda a cada Parte que tome las medidas necesarias para tipificar como delito en su derecho interno cada uno de los apartados descritos en cada categoría.

En la Disposición 14221 del BOE núm. 226 de 2010 se encuentra el Instrumento de Ratificación del Convenio sobre la Ciberdelincuencia, hecho en Budapest el 23 de noviembre de 2001.

### ***4.3 Normativa y regulación de la informática en el ámbito nacional***

#### **4.3.1 Leyes y Decretos Ley**

##### **4.3.1.1 Ley Orgánica de Protección de datos de carácter personal.**

Régimen sancionador aplicable (BOE No298 de 14/XII/99 que publicó la Ley Org. 15/1999 de 13 de Dic.).

**Objeto:** Proteger y garantizar las libertades públicas y derechos fundamentales de las personas, especialmente su HONOR e INTIMIDAD personal y familiar.

**Aspectos de interés:** Serán responsables: “Los responsables de los ficheros o de los tratamientos” y “los encargados de los tratamientos”.

##### **Tipos de infracciones:**

- Leves (art.44.2): multas de 100.000 a 10M pts.

Ej. Rectificar datos o no comunicarlos a la Agencia de Protección de Datos.

- Graves (art.43): multas de 10M a 50M pts.

Ej. No mantener sistemas de seguridad, obstrucción o inspección, uso en provecho propio...

- Muy Graves (art.45): multas de 50M a 100M pts (“Conductas reprochables”).

Ej. Vulnerar a propósito el secretismo, etc...

#### **4.3.1.2 Ley 7/1998 de 13 de abril que regula las condiciones generales de contratación.**

R.D. 1906/1999 de 17/XII que regula la contratación telefónica.

R.D. Ley 14/1999 de 17/XII sobre Firma Electrónica (BOE No224 de 18/XII).

- **Firma electrónica:** Dispositivo electrónico que permite la identificación del signatario de las operaciones realizadas por Internet.
- **Identifica:** El firmante (autenticación) y evita el retracto (no repudio).

### **4.3.2 Código Penal**

#### **4.3.2.1 Ley Orgánica 10/1995 de 23/XI**

Tipifica delitos y faltas por el uso de la informática, concretamente contra la Intimidad, Patrimonio, Socioeconómicos y Propiedad Intelectual.

**Título X:** “Delitos contra la intimidad, derecho a la propia imagen y la inviolabilidad del Domicilio”.

- Apoderarse de papeles, e-mails, mensajes, otros...
- Cracks: delitos.
- Obtener datos de terceros...



### **4.3.3 Recomendaciones de la APD**

#### **4.3.3.1 Información en la recogida de datos**

- Cuando suministre datos personales a cualquier organización (proveedores de acceso, proveedores de contenido, vendedores a través de comercio electrónico, etc.) sea consciente de a quién se los facilita y con qué finalidad.
- Procure averiguar la política de sus proveedores y administradores de listas y directorios en lo que se refiere a venta, intercambio o alquiler de los datos que les suministra. Solicite que sus datos personales no vayan unidos a su identificación de acceso a Internet.

#### **4.3.3.2 Finalidad para la que se recogen los datos**

- Desconfíe si los datos que le solicitan son excesivos para la finalidad con la que se recogen o innecesarios para el servicio que se le presta.
- Tenga en cuenta que cuando introduce su dirección de correo electrónico en un directorio, lista de distribución o grupo de noticias, dicha dirección puede ser recogida por terceros para ser utilizada con una finalidad diferente, como por ejemplo, remitirle publicidad no deseada.
- Cuando navegue por Internet, sea consciente de que los servidores Web que visita pueden registrar tanto las páginas a las que accede como la frecuencia y los temas o materias por las que busca, aunque no le informen de ello. Asimismo, su pertenencia a determinados grupos de noticias y listas de distribución puede contribuir a la elaboración de perfiles más o menos detallados sobre su persona. En el caso de que no desee dejar constancia de sus actividades en la red, utilice los mecanismos para preservar el anonimato que se describen en el cuerpo de este documento.

#### **4.3.3.3 Seguridad en el intercambio de datos**

- Utilice, siempre que sea posible, las últimas versiones de los programas navegadores, ya que cada vez suelen incorporar mejores medidas de seguridad. Considere la posibilidad de activar en dichos programas las opciones que alerten sobre los intercambios de datos no deseados y no rellene aquellos datos que no

desea hacer públicos (por ejemplo, dirección de correo electrónico, nombre, apellidos, etc.).

- No realice transacciones comerciales electrónicas a través de proveedores con sistemas inseguros o no fiables. Consulte el manual de su navegador para averiguar cómo informa de que se ha establecido una conexión con un servidor seguro.
- Recuerde que existen sistemas de dinero electrónico que preservan el anonimato de sus compras en Internet.
- Utilice los mecanismos de seguridad que tenga a su alcance para proteger sus datos de accesos no deseados. El medio más fiable para conseguirlo es el cifrado de los mismos.
- Salvo que se utilicen mecanismos de integridad, autenticación y certificación (firma digital, notarios electrónicos, etc.) no confíe ciegamente en que la persona u organización que le remite un mensaje es quien dice ser y en que el contenido del mismo no se ha modificado, aunque esto sea así en la inmensa mayoría de las ocasiones.

#### **4.3.3.4 Para terminar**

- Siempre que se le soliciten datos personales que no esté obligado legalmente a suministrar, sopesa los beneficios que va a recibir de la organización que los recoge frente a los posibles riesgos de utilización irregular de los mismos.
- Ante cualquier duda sobre la legalidad de la utilización de sus datos de carácter personal, póngase en contacto con la Agencia de Protección de Datos.

### ***4.4 La protección jurídica de programas de ordenador. Piratería informática***

- El Real Decreto Legislativo 1/1996, por el que se aprueba el Texto Refundido sobre Propiedad Intelectual, la protección jurídica de los programas de ordenador, antes regulada por la Ley de Protección Jurídica de Programas de Ordenador y por

la Ley de Propiedad Intelectual, crea un marco jurídico en contra de la piratería informática.

- El Texto Refundido desarrolla una serie de medidas para combatir la piratería informática, como la posibilidad de que los fabricantes de programas de ordenador soliciten a la justicia española la realización de un registro sorpresa en empresas en las que existan sospechas fundadas o evidencias de delito. España es uno de los países en los que se puede acudir a esta medida cautelar. De esta manera se erradica la posibilidad de que los presuntos infractores puedan destruir las pruebas existentes lo cual, indudablemente, ocurriría si se les notificase por adelantado la realización de un registro.

#### **4.4.1 ¿En qué casos se infringe la Ley?**

- Al copiar o distribuir un programa de ordenador o la documentación que le acompaña, incluidas aplicaciones, datos, códigos y manuales, sin permiso expreso o licencia del propietario de los derechos de explotación.
- Al utilizar un programa sin la correspondiente licencia o autorización del fabricante, con independencia de que se utilice en un solo ordenador o en varios de forma simultánea.
- Al utilizar programas de ordenador en un número de copias superior al autorizado por el fabricante en sus contratos o licencias de uso.
- En empresas y demás organizaciones, al fomentar, consciente o inconscientemente, permitir, obligar o presionar a los empleados a realizar o distribuir copias no autorizadas del programa.
- Al efectuar copias no autorizadas porque alguien lo requiere u obliga a ello. Al ceder o prestar el programa de forma que pueda ser copiado o al copiarlo mientras está en su posesión en calidad de cedido o prestado.

- Al crear, importar, poseer o negociar con artículos destinados a burlar o neutralizar cualquier medio técnico aplicado para proteger el programa de ordenador.

#### **4.4.2 Medidas Judiciales**

Si finalmente existe evidencia de delito, las medidas judiciales que pueden adoptarse son:

- Solicitar al Juez un registro sorpresa de las instalaciones del presunto infractor, tanto por la vía civil, como por la penal.
- Solicitar al Juez la adopción urgente de medidas cautelares de protección.
- Exigir indemnizaciones acordes con los daños materiales y morales causados.
- El cierre del centro de actividad del infractor.
- El secuestro de todos aquellos medios destinados a suprimir los dispositivos técnicos que protegen un programa desarrollado y comercializado por un fabricante de programas.

#### **4.5 Normativa y regulación de tallas mínimas**

En el ámbito legislativo comunitario de las tallas mínimas [7], la norma en vigor es el Reglamento (CE) nº 850/1998 del Consejo, de 30 de marzo de 1998, para la conservación de los recursos pesqueros a través de medidas técnicas de protección de los juveniles de organismos marinos. Este Reglamento establece las tallas mínimas aplicables en cada una de las regiones a las que hace referencia. Para España abarca la Región 3, es decir, las aguas correspondientes a las zonas CIEM VIII y IX; así como la Región 5 que recoge las aguas situadas en la parte del Atlántico centro-oriental (las divisiones 34.1.1, 34.1.2 y 34.1.3; y la subzona 34.2.0 de la zona de pesca 34 de la región del CPACO).

Posteriormente, se publica el Reglamento (CE) nº 1967/2006 del Consejo, de 21 de diciembre de 2006, relativo a las medidas de gestión para la explotación sostenible

de los recursos pesqueros en el Mar Mediterráneo y por el que se modifica el Reglamento (CEE) nº 2847/1993 y se deroga el Reglamento (CE) nº 1626/1994. Este Reglamento establece las tallas mínimas de desembarque de determinados organismos marinos, con el doble objetivo de mejorar su explotación y de fijar normas a partir de las cuales los Estados miembros puedan construir su sistema de gestión para la pesca de bajura.

En 2007 se publica el Reglamento (CE) nº 520/2007 del Consejo, de 7 de mayo de 2007, por el que se establecen medidas técnicas de conservación de determinadas poblaciones de peces de especies altamente migratorias y por el que se deroga el Reglamento (CE) nº 973/2001. Este Reglamento, aunque parcialmente derogado, continua vigente en lo relativo a la talla mínima del pez espada.

En 2009 entra en vigor el Reglamento (CE) nº 302/2009 del Consejo, de 6 de abril de 2009, por el que se establece un plan de recuperación plurianual para el atún rojo del Atlántico oriental y el Mediterráneo, se modifica el Reglamento (CE) nº 43/2009 y se deroga el Reglamento (CE) nº 1559/2007. En él se establece una nueva talla mínima para el atún rojo.

Situándonos ahora en el ordenamiento nacional, en 1995 entra en vigor el Real Decreto 560/1995, de 7 de abril, por el que se establece las tallas mínimas de determinadas especies pesqueras. Con esta norma se regula a escala nacional la obligación de respetar tallas concretas en determinadas especies de productos pesqueros.

Posteriormente se modifica mediante el Real Decreto 1615/2005, de 30 de diciembre, por el que se establecen las tallas mínimas de determinadas especies pesqueras para cada uno de los caladeros nacionales.

En 2006, se publica la Orden APA/2521/2006, de 27 de julio, por la que se regula la pesca con el arte de palangre de superficie para la captura de especies altamente migratorias y por la que se crea el censo unificado de palangre de superficie. Mediante esta Orden se establece la talla mínima del pez espada para cada una de las zonas de pesca.

En definitiva, en el ámbito nacional, las exigencias para cada uno de los caladeros y por especie quedan claramente definidas:

- . Caladero del Archipiélago Canario.
- . Caladero del Mediterráneo.
- . Caladero del Cantábrico, Noroeste y del Golfo de Cádiz.

Adicionalmente, y para cada especie pesquera, se debe consultar la normativa comunitaria, así como la autonómica vigente.

#### 4.5.1 Legislación Comunitaria

- **Reglamento (CE) nº 850/98 del Consejo, de 30 de marzo de 1998** para la conservación de los recursos pesqueros a través de medidas técnicas de protección de los juveniles de organismos marinos.

 [Reglamento \(CE\) 850/98 del Consejo \(340 Kb\)](#)

- **Reglamento (CE) nº 1967/2006 del Consejo, de 21 de diciembre de 2006** relativo a las medidas de gestión para la explotación sostenible de los recursos pesqueros en el Mar Mediterráneo y por el que se modifica el Reglamento (CEE) nº 2847/93 y se deroga el Reglamento (CE) nº 1626/94.

 [Reglamento \(CE\) 1967/2006 del Consejo \(774 Kb\)](#)

- **Reglamento (CE) nº 520/2007 del Consejo, de 7 de mayo de 2007** por el que se establecen medidas técnicas de conservación de determinadas poblaciones de peces de especies altamente migratorias y por el que se deroga el Reglamento (CE) nº 973/2001.

 [Reglamento \(CE\) 520/2007 del Consejo \(87 Kb\)](#)

- **Reglamento (CE) nº 302/2009 del Consejo, de 6 de abril de 2009** por el que se establece un plan de recuperación plurianual para el atún rojo del Atlántico oriental y el Mediterráneo, se modifica el Reglamento (CE) nº 43/2009 y se deroga el Reglamento (CE) nº 1559/2007.

 [Reglamento \(CE\) 302/2009 del Consejo \(1046 Kb\)](#)

## 4.5.2. Legislación Nacional

- **Real Decreto 1615/2005, de 30 de diciembre** por el que se modifica el Real Decreto 560/1995, de 7 de abril, por el que se establecen las tallas mínimas de determinadas especies pesqueras para cada uno de los caladeros nacionales.

 [Real Decreto 1615/2005 \(74 Kb\)](#)

- **Orden APA/2521/2006, de 27 de julio** por la que se regula la pesca con el arte de palangre de superficie para la captura de especies altamente migratorias y por la que se crea el censo unificado de palangre de superficie.

## 4.6 GNU General Public License

La Licencia Pública General de GNU o más conocida por su nombre en inglés GNU General Public License o simplemente sus siglas del inglés GNU GPL, es una licencia creada por la Free Software Foundation en 1989 (la primera versión), y está orientada principalmente a proteger la libre distribución, modificación y uso de software. Su propósito es declarar que el software cubierto por esta licencia es software libre y protegerlo de intentos de apropiación que restrinjan esas libertades a los usuarios.

Existen varias licencias "hermanas" de la GPL, como la licencia de documentación libre de GNU (GFDL), la Open Audio License, para trabajos musicales, etcétera, y otras menos restrictivas, como la LGPL, o la LGPL (Lesser General Public License, antes Library General Public License), que permiten el enlace dinámico de aplicaciones libres a aplicaciones no libres. Android y Java fueron liberados bajo esta licencia.

La licencia GPL, al ser un documento que cede ciertos derechos al usuario, asume la forma de un contrato, por lo que usualmente se la denomina contrato de licencia o acuerdo de licencia.

## **4.7. Software Libre**

El software libre (en inglés free software, aunque esta denominación también se confunde a veces con "gratis" por la ambigüedad del término en el idioma inglés) es la denominación del software que respeta la libertad de los usuarios sobre su producto adquirido y, por tanto, una vez obtenido puede ser usado, copiado, estudiado, modificado y redistribuido libremente. Según la Free Software Foundation, el software libre se refiere a la libertad de los usuarios para ejecutar, copiar, distribuir, estudiar, modificar el software y distribuirlo modificado.

El software libre suele estar disponible gratuitamente, o al precio de costo de la distribución a través de otros medios; sin embargo no es obligatorio que sea así, por lo tanto no hay que asociar software libre a "software gratuito" (denominado usualmente freeware), ya que, conservando su carácter de libre, puede ser distribuido comercialmente ("software comercial"). Análogamente, el "software gratis" o "gratuito" incluye en ocasiones el código fuente; no obstante, este tipo de software no es libre en el mismo sentido que el software libre, a menos que se garanticen los derechos de modificación y redistribución de dichas versiones modificadas del programa.

Tampoco debe confundirse software libre con "software de dominio público". Éste último es aquel software que no requiere de licencia, pues sus derechos de explotación son para toda la humanidad, porque pertenece a todos por igual. Cualquiera puede hacer uso de él, siempre con fines legales y consignando su autoría original. Este software sería aquel cuyo autor lo dona a la humanidad o cuyos derechos de autor han expirado, tras un plazo contado desde la muerte de este, habitualmente 70 años. Si un autor condiciona su uso bajo una licencia, por muy débil que sea, ya no es del dominio público.

## **4.8. Licencia de software**

Una licencia de software otorga al usuario derecho legal a utilizar un software. Por cada programa de software de Microsoft que se utiliza, se otorga una licencia al usuario y ésta se documenta en el Contrato de Licencia de Usuario Final (CLUF). Un usuario de software, necesita una licencia. El acuerdo de licencia da al usuario el derecho de utilizar el software. El software está protegido por la ley de derechos de autor, que establece que el producto no se puede copiar sin autorización del dueño de



derechos de autor. Hay maneras diferentes de adquirir una licencia de Software Microsoft:

- Producto Empaquetado (Caja): Licencia, CD-Rom y documentación en un paquete.

- Original Equipment Manufacturer (OEM): licencia para software preinstalado en un PC nuevo.

- Licencia por Volumen.

## 5. DESARROLLO

### 5.1 Metodología

La metodología software que se ha empleado para el desarrollo del proyecto ha sido el modelo en cascada, ya que conocíamos todas las especificaciones desde un principio y es un modelo fácil de planificar.

El modelo en cascada [15], es el enfoque metodológico que ordena rigurosamente las etapas del proceso para el desarrollo de software, de tal forma que el inicio de cada etapa debe esperar a la finalización de la etapa anterior.

Este modelo consta normalmente de las siguientes etapas principales de creación:

1. Análisis de requisitos
2. Diseño
3. Codificación
4. Prueba
5. Mantenimiento

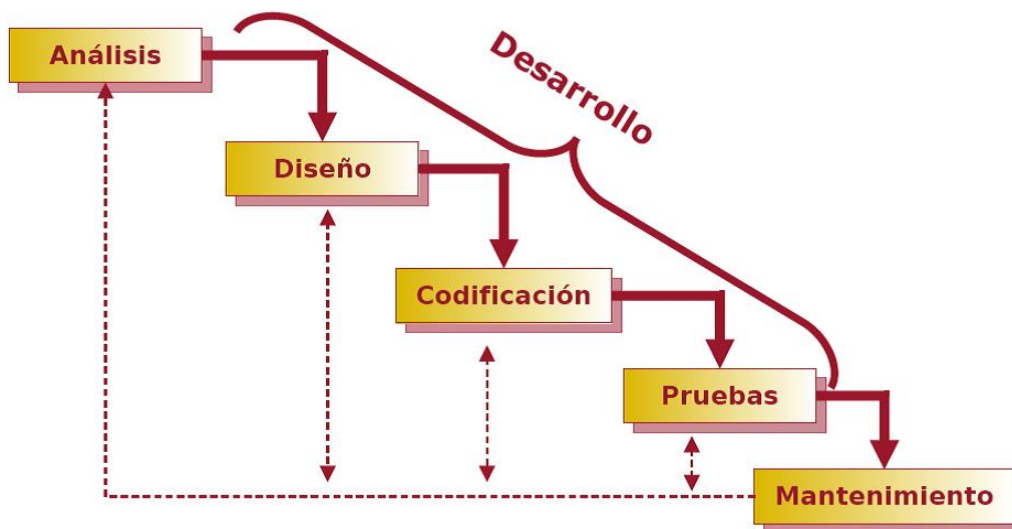


Figura 5.1- Fases del modelo en cascada [16].

Este modelo admite la posibilidad de hacer iteraciones, es decir, durante las modificaciones que se hacen en el mantenimiento se puede ver por ejemplo la necesidad de cambiar algo en el diseño, lo cual significa que se harán los cambios necesarios en la codificación y se tendrán que realizar de nuevo las pruebas, es decir, si se tiene que volver a una de las etapas anteriores al mantenimiento hay que recorrer de nuevo el resto de las etapas.

## **5.2 Etapas del trabajo**

El desarrollo de trabajo consta de varias etapas diferenciadas, desde el inicio hasta el final del proyecto. Cada una ha sido fundamental y dependiente de la anterior, aportando solidez y estructura al contenido del mismo. Dichas etapas son las que se enumeran a continuación:

### ➤ **Análisis de requisitos**

En esta etapa se estudiará la documentación relacionada con la idea a llevar a cabo el análisis de las aplicaciones más significativas y relacionadas con el proyecto, así como los recursos de software necesarios para el desarrollo de la misma.

### ➤ **Diseño del proyecto**

A la vez que se realizan los aspectos fundamentales de las funcionalidades de la aplicación hay que ir dotando de diseño al proyecto para poder tener una idea clara de lo que se está haciendo; siempre primando la facilidad de uso y una interfaz amigable.

### ➤ **Implementación del proyecto**

Una vez concluidas las etapas anteriores procedemos a la implementación del prototipo de la aplicación. Para ello es necesario realizar dos fases bien diferenciadas:

- Entrenamiento de los detectores.
- Desarrollo del software prototipo de detección.

## ➤ Prueba y mantenimiento del proyecto

Se han realizado pruebas desde el inicio hasta el final de la implementación del mismo, cumpliendo con las especificaciones, las cuales han sido testeadas en todo momento.

Las primeras pruebas han sido realizadas sobre los clasificadores, probándolos en diversas condiciones de luminosidad y a diferentes distancias respecto al objeto a detectar, comprobando así su efectividad y realizando las mejoras necesarias. Posteriormente se realizaron pruebas sobre la detección simultánea del pez y de la moneda.

Una vez terminada la codificación se ha realizado el mantenimiento del mismo en busca de mejoras y posibles errores, probando el prototipo sobre diferentes especies de varios tamaños y en diversas situaciones.

### 5.2.1 Análisis de requisitos

El desarrollo de este proyecto de software precisa de un análisis previo de las características del aplicativo.

El objetivo del proyecto, como hemos mencionado anteriormente, se basa en el desarrollo de una aplicación que detecte un pez y nos indique si cumple la talla mínima, la cual se desarrollará sobre una plataforma de móvil, concretamente hemos decidido realizarla sobre la plataforma Android.

#### 5.2.1.1 Esquema de Viola-Jones para la detección de objetos.

El sistema de detección de objetos desarrollado por Viola-Jones [3] [4] es el primer sistema de detección de objetos en ofrecer tasas competitivas para la detección de objetos en tiempo real.

El esquema de Viola-Jones supuso un notable avance al conseguir reducir los tiempos de ejecución de los esquemas de detección basados en ventana deslizante.

La idea básica radica en la combinación en un esquema en cascada de clasificadores débiles en lugar de un único clasificador más potente. La combinación en cascada demuestra un rendimiento similar, al mismo tiempo que dedica menos procesamiento a zonas de la imagen poco prometedoras que son rechazadas en las primeras etapas de la cascada, obteniendo un menor tiempo de procesamiento.

A pesar de que pueden ser entrenados para detectar una variedad de clases de objetos, ha sido muy divulgado dada su efectividad en la detección de rostros.

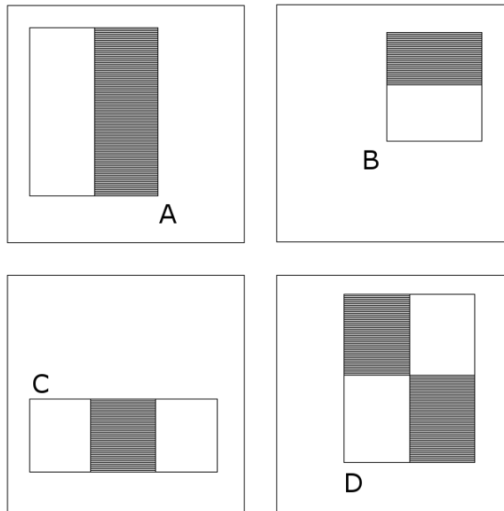
#### ▪ **Las características Haar**

La principal razón para usar características en el algoritmo es que permite hacer una asociación entre conocimiento aprendido y el análisis de los datos adquiridos. Además, la clasificación por características es mucho más compacta que el procesado por análisis basados en píxel.

Las características usadas son las mismas que fueron usadas por Papageorgiou et al. [8]. Las características de Haar (Haar-like features en inglés) permiten obtener información de una zona concreta mediante una operación aritmética simple: Esto nos lleva a una gran eficiencia tanto de cálculo como espacial.

En concreto se usan tres características de Haar:

- Característica de dos rectángulos.
- Característica de tres rectángulos.
- Característica de cuatro rectángulos.



**Figura 5.2-** Figura que muestra el tipo y la configuración de los detectores de características utilizado por el Marco de detección de objetos en tiempo real robusto Viola -Jones

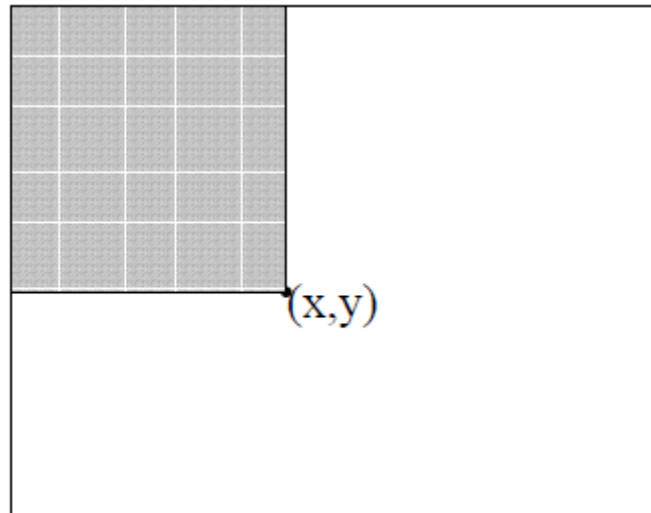
En todos los casos anteriores la obtención del valor de la característica consiste en la resta entre el valor de una o más subzonas dentro de la zona analizada realizándose un análisis para un alto número de combinaciones. Es decir, restamos el valor que damos una subzona, mediante la integral sumada (explicada más adelante en este mismo documento), con el de otra subzona.

- **Imagen integral**

Las características Haar basadas en zonas rectangulares se pueden calcular muy rápidamente usando una representación intermedia de la imagen llamada “imagen integral”. La imagen integral en el lugar  $(x,y)$  contiene la suma de los píxeles situados arriba y a la izquierda de  $(x,y)$ :

$$ii(x, y) = \sum_{x' \leq x, y' \leq y} i(x', y'),$$

Donde  $ii(x,y)$  es la imagen integral y la imagen original es  $i(x,y)$ , siendo  $x$  e  $y$  las coordenadas de un píxel.



**Figura 5.2-** El valor de la imagen integral en el punto  $(x,y)$  es la suma de todos los píxeles situados arriba y a la izquierda.

La imagen integral representa en computación una manera elaborada de obtener valores de forma eficiente. Este método permite el uso de programación dinámica, que permite la obtención de valores dentro de la imagen a través de otros valores calculados previamente.

- **AdaBoost**

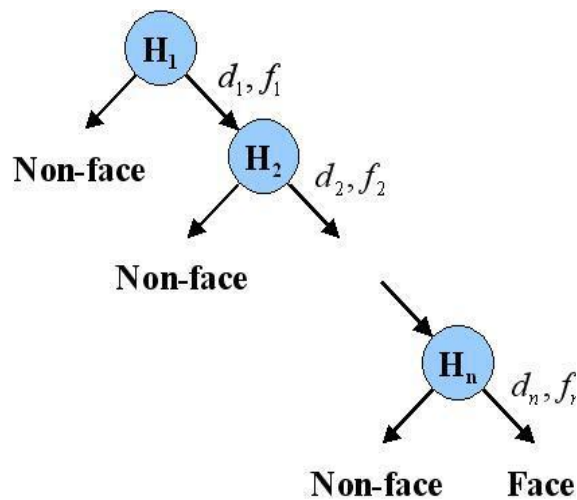
AdaBoost (Adaptative Boost) es un algoritmo de aprendizaje automático que consiste en la clasificación de características por medio de un grupo de clasificadores. El algoritmo define un criterio para buscar la mejor forma de aplicar esos clasificadores para detectar favorablemente el objeto de interés.

En el algoritmo de Viola-Jones el algoritmo AdaBoost es capaz de elegir entre un gran conjunto de filtros, las características de Haar, para seleccionar en cada momento cuál de ellos se ajusta mejor para el problema de detección particular que nos ocupe.

- **Arquitectura en cascada**

El algoritmo de Viola-Jones utiliza un árbol binario de clasificadores para decidir si una región se trata de una cara o no. Esta cascada está formada por nodos en los

que se analizan las diferentes características de las diferentes regiones de la imagen. Cada nodo representa un clasificador obtenido con el algoritmo AdaBoost.



*Figura 5.4- Árbol binario de clasificadores (fuente: es.wikipedia.org)*

El proceso de aprendizaje que utiliza el algoritmo de Viola-Jones permite como valor de entrada un error que permitiremos en la clasificación en el primer nodo del árbol.

Utilizando ese valor lo que intenta el algoritmo es crear un clasificador que en el primer nodo muestre el error máximo que hemos permitido en la detección y según el árbol se vuelve más profundo el criterio con el que descartamos es mucho más estricto. De esta manera cuando una muestra llega al último nivel del árbol sabemos que ha pasado por todos los niveles de detección.

Además, esta forma de organizar el árbol de decisión nos permite descartar en un primer momento las muestras más sencillas de rechazar, es decir aquellas que claramente no se corresponden con el objeto buscado, y aplicar un mayor esfuerzo a aquellas cuya clasificación sea más dudosa.

### 5.2.1.2 OpenCV

OpenCV es una librería de visión por computador de código abierto [2]. Está escrito en C y C++ y corre bajo Linux, Windows y Mac OS X. Existen también interfaces para Python, Ruby, Matlab y otros lenguajes.



OpenCV ha sido diseñado para la eficiencia computacional y con un fuerte enfoque en aplicaciones de tiempo real. Está escrito en lenguaje optimizado y puede aprovechar los procesadores multinúcleo.

Uno de los objetivos de OpenCV es proporcionar una infraestructura de visión por computador fácil de utilizar, que ayuda a crear rápidamente aplicaciones de visión bastante sofisticadas. La librería OpenCV contiene más de 500 funciones que abarcan muchas áreas de la visión, incluyendo el reconocimiento de productos de fábricas, imágenes médicas, seguridad, interfaz de usuario, calibración de cámara, visión estéreo y robótica. Debido a que la visión por computador y el aprendizaje automático aparecen a menudo juntos, OpenCV contiene también una completa librería de Aprendizaje Automático o, en inglés, Machine Learning Library (MLL). Esta librería se centra en el reconocimiento de patrones estadísticos y de agrupamiento. La MLL es de gran utilidad para las tareas de visión que se encuentran en el núcleo de la misión de OpenCV, pero es lo suficientemente general como para ser utilizada para cualquier problema de aprendizaje automático.

#### ▪ **Detección en OpenCV**

OpenCV integra una implementación del esquema de Viola-Jones, facilitándonos la tarea de detección. Actualmente cuenta tanto con clasificadores entrenados para esta tarea almacenados en archivos xml, como con las herramientas necesarias para poder crear nuestros propios clasificadores en caso de que los necesitemos.

Para detectar un objeto debemos cargar la imagen en la cual deseamos buscarlo, luego de cargar la imagen debemos aplicar los siguientes pasos:

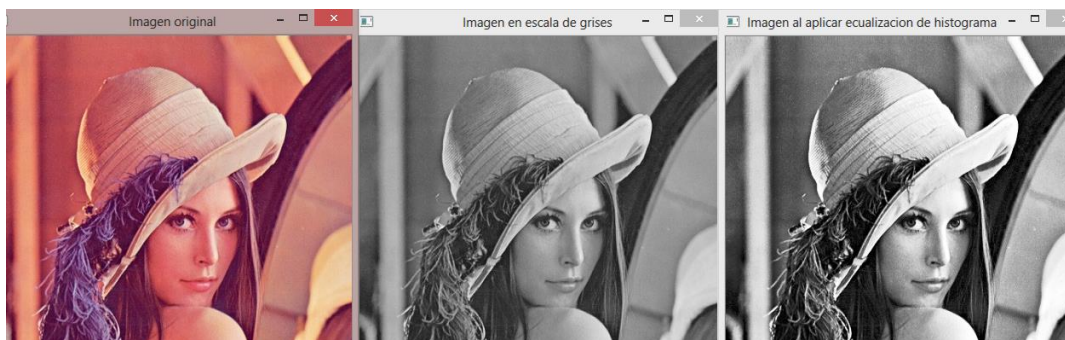
1. Convertir la imagen a escala de grises, necesario para el correcto funcionamiento de los algoritmos de detección de rostros usados por OpenCV. Para convertir una imagen a escala de grises u otro formato contamos con la función *cvtColor*.

```
cvtColor(imagen, imagen, CV_BGR2GRAY);
```

2. Lo siguiente que debemos hacer es ecualizar el histograma de la imagen en grises para estandarizar el contraste y brillo de la imagen, para que las distintas condiciones de iluminación no afecten la detección del objeto de interés en la imagen.

```
equalizeHist(imagen, imagen);
```

Con esto tendremos este resultado:



**Figura 5.5-** Resultado de los procesos de transformación de las imágenes.

Para aumentar la velocidad de detección del algoritmo podemos escalar la imagen a un tamaño más pequeño sin exceder el tamaño mínimo del patrón buscado.

3. Una vez realizado el preprocesamiento de la imagen, procederemos a cargar el clasificador, si no estuviera ya disponible, pasándole el nombre del clasificador al método *load* de la clase *CascadeClassifier*, los archivos .xml que debemos cargar se encuentran en la carpeta *data/haarcascades* de la distribución de OpenCV. Aquí encontraremos varias carpetas que contienen distintos clasificadores, en la carpeta *data/haarcascades* se encuentran varios clasificadores no solo para detectar rostros sino también para la detección de ojos, boca, nariz, entre otros.

Por ejemplo para detectar rostros de frente usaremos *haarcascade\_frontalface\_alt.xml*, para detectar cuerpo completo podemos usar *haarcascade\_fullbody.xml*, para detectar ojos contamos con *haarcascade\_eye.xml*, existen muchos otros.

```
CascadeClassifier detector;
if(!detector.load("haarcascade_frontalface_alt.xml"))
    cout << "No se puede abrir clasificador."<< endl;
```

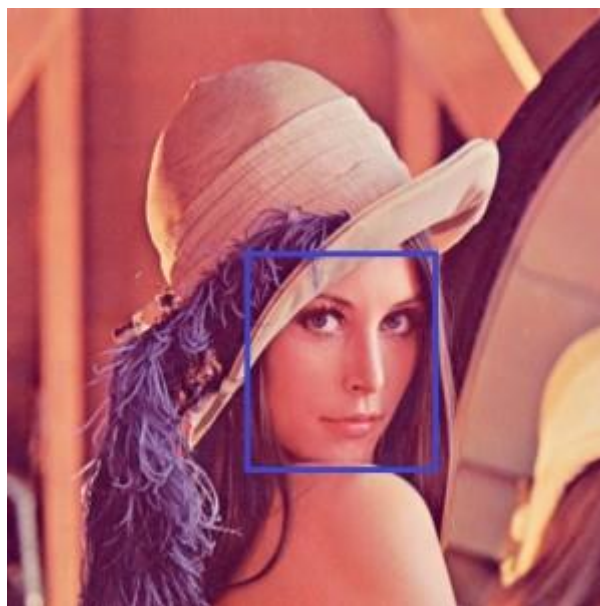
#### 4. Ahora podemos buscar los objetos presentes en la imagen

```
vector<Rect> rect;
detector.detectMultiScale(dest, rect);
```

Los objetos detectados serán almacenados en el vector rect, solo debemos recorrerlo y dibujar los contenedores de los encontrados.

```
for(Rect rc : rect)
{
    rectangle(imagen,
        Point(rc.x, rc.y),
        Point(rc.x + rc.width, rc.y + rc.height),
        CV_RGB(0,255,0), 2);
```

En la Figura 5.6 podemos observar el dibujo que indica la detección de un rostro.



**Figura 5.6-** Localización y Segmentación de un rostro con Viola-Jones [21].

```

1 #include <opencv2\opencv.hpp>
2
3 using namespace cv;
4 using namespace std;
5
6 int main()
7 {
8     Mat dest, gray;
9     Mat imagen = imread("lena.jpg");
10
11     CascadeClassifier detector;
12
13     if(!detector.load("haarcascade_frontalface_alt.xml"))
14         cout << "No se puede abrir clasificador." << endl;
15
16     cvtColor(imagen, gray, CV_BGR2GRAY);
17     equalizeHist(gray, dest);
18
19     vector<Rect> rect;
20     detector.detectMultiScale(dest, rect);
21
22     for(Rect rc : rect)
23     {
24         rectangle(imagen,
25                 Point(rc.x, rc.y),
26                 Point(rc.x + rc.width, rc.y + rc.height),
27                 CV_RGB(0,255,0), 2);
28     }
29
30     imshow("Imagen original", imagen);
31     imshow("Imagen en escala de grises", gray);
32     imshow("Imagen al aplicar ecualizacion de histograma",dest);
33
34     waitKey(0);
35     return 1;
36 }

```

Figura 5.7-Código completo en c++.

- **Detección en tiempo real**

Para detectar objetos en tiempo real solo debemos cargar las imágenes de la cámara y aplicar la función *detectMultiscale*, la cual devuelve los objetos encontrados como una lista.

Parámetros de la función de detección:

```
detectMultiScale(const Mat& image, vector<Rect>& objects, double  
scaleFactor=1.1, int minNeighbors=3, int flags=0, Size minSize=Size(), Size  
maxSize=Size())
```

- **image:** imagen a procesar.
- **objects:** vector donde se almacenara las coordenadas de los objetos encontrados.
- **scaleFactor:** factor de escala, 1.2 más rápido.
- **minNeighbors:** factor para prevenir falsas detecciones.
- **flags:** parámetro actualmente sin uso, indicar 0.
- **minSize, maxSize:** tamaño máximo y mínimo de ventana usado para la detección, limita la detección de objetos muy pequeños o demasiado grandes.

### 5.2.1.3 Comparativa entre plataformas móviles.

A lo largo de estos años, los fabricantes se han basado en distintos sistemas operativos para programar y gestionar los distintos terminales, del mismo modo, que los ordenadores han utilizado sistemas operativos como Windows, Unix o MAC OS.

Los sistemas operativos de los teléfonos móviles son muchos más simples y fundamentalmente están orientados a expresar las capacidades multimedia e inalámbricas de los dispositivos.

Del mismo modo que los teléfonos móviles han ido creciendo en popularidad, los sistemas operativos han ido adquiriendo una mayor importancia, existiendo una guerra por hacerse con la mayor cuota de mercado. Si hace unos años los principales motivos para la elección de un teléfono móvil eran la duración de batería y un diseño atractivo, ahora el sistema operativo se ha convertido en uno de los principales factores a la hora de decantarnos por uno.

En este apartado vamos a describir las características de las principales plataformas móviles disponibles en la actualidad. Las plataformas comparadas y la versión que se ha utilizado como referencia se muestran en la Figura 5.8.



Figura. 5.8 - Plataformas móviles comparadas.

En la Figura 5.9 vemos reflejadas las principales características de cada una de ellas:

	Apple iOS 7	Android 4.3	Windows Phone 8	BlackBerry OS 7	Symbian 9.5
Compañía	Apple	Open Handset Alliance	Microsoft	RIM	Symbian Foundation
Núcleo del SO	Mac OS X	Linux	Windows NT	Mobile OS	Mobile OS
Licencia de software	Propietaria	Software libre y abierto	Propietaria	Propietaria	Software libre
Año de lanzamiento	2007	2008	2010	2003	1997
Fabricante único	Sí	No	No	Sí	No
Variedad de dispositivos	modelo único	muy alta	media	baja	muy alta
Soporte memoria externa	No	Sí	Sí	Sí	Sí
Motor del navegador web	WebKit	WebKit	Pocket Internet Explorer	WebKit	WebKit
Soporte Flash	No	Sí	No	Si	Sí
HTML5	Sí	Sí	Sí	Sí	No
Tienda de aplicaciones	App Store	Google Play	Windows Marketplace	BlackBerry App World	Ovi Store
Número de aplicaciones	825.000	850.000	160.000	100.000	70.000
Coste publicar	\$99 / año	\$25 una vez	\$99 / año	sin coste	\$1 una vez
Actualizaciones automáticas del S.O.	Sí	depende del fabricante	depende del fabricante	Sí	Sí
Familia CPU soportada	ARM	ARM, MIPS, Power, x86	ARM	ARM	ARM
Máquina virtual	No	Dalvik	.net	Java	No
Aplicaciones nativas	Siempre	Sí	Sí	No	Siempre
Lenguaje de programación	Objective-C, C++	Java, C++	C#, muchos	Java	C++
Plataforma de desarrollo	Mac	Windows, Mac, Linux	Windows	Windows, Mac	Windows, Mac, Linux

Figura 5.9- Comparativa de las principales plataformas móviles. (fuente: <http://www.androidcurso.com/>)

Otro aspecto fundamental a la hora de comparar las plataformas móviles es su cuota de mercado. En la Figura 5.10 podemos ver un estudio realizado por la empresa

Gratner Group [9], donde se muestra la evolución del mercado de los sistemas operativos para móviles según el número de terminales vendidos. Podemos destacar: el importante descenso de ventas de la plataforma Symbian de Nokia; el declive continuo de BlackBerry; como la plataforma de Windows que parece que no despega; como Apple tiene afianzada una cuota de mercado en torno al 15%. Finalmente destacamos el espectacular ascenso de la plataforma Android, que le ha permitido alcanzar en dos años una cuota de mercado superior al 75%.

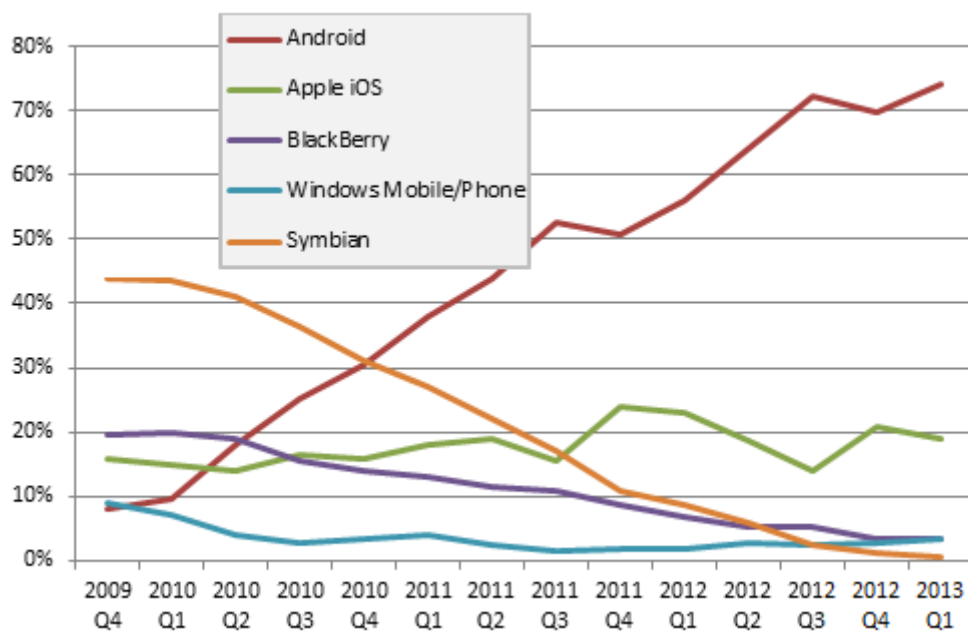


Figura. 5.10- Porcentaje de teléfonos inteligentes vendidos según su sistema operativos hasta el tercer cuarto del 2013 en el mundo [9].

#### 5.2.1.4 Plataforma de trabajo: Android

Existen muchas plataformas para móviles, sin embargo Android [10] [11] presenta una serie de características que lo hacen diferente, por las que lo hemos seleccionado para realizar este proyecto:

- **Plataforma realmente abierta.** Es una plataforma de desarrollo libre basada en Linux y de código abierto. Una de sus grandes ventajas es que se puede usar y adaptar el sistema sin pagar royalties.
- **Adaptable a cualquier tipo de hardware.** Android no ha sido diseñado exclusivamente para su uso en teléfonos y tabletas. Hoy en día podemos encontrar

relojes, cámaras, electrodomésticos y gran variedad de sistemas empotrados que se basan en este sistema operativo. Este hecho tiene sus evidentes ventajas, pero también va a suponer un esfuerzo adicional al programador. La aplicación ha de funcionar correctamente en dispositivos con gran variedad de tipos de entrada, pantalla, memoria, etc. Esta característica contrasta con la estrategia de Apple. En iOS tenemos que desarrollar una aplicación para iPhone y otra diferente para iPad.

- **Portabilidad asegurada.** Las aplicaciones finales son desarrolladas en Java lo que nos asegura que podrán ser ejecutadas en cualquier tipo de CPU, tanto presente como futuro. Esto se consigue gracias al concepto de máquina virtual.
- **Arquitectura basada en componentes inspirados en Internet.** Por ejemplo, el diseño de la interfaz de usuario se hace en xml, lo que permite que una misma aplicación se ejecute en un móvil de pantalla reducida o en un TV.
- **Filosofía de dispositivo siempre conectado a Internet.**
- **Gran cantidad de servicios incorporados.** Por ejemplo, localización basada tanto en GPS como en redes, bases de datos con SQL, reconocimiento y síntesis de voz, navegador, multimedia.
- **Aceptable nivel de seguridad.** Los programas se encuentran aislados unos de otros gracias al concepto de ejecución dentro de una caja que hereda de Linux. Además, cada aplicación dispone de una serie de permisos que limitan su rango de actuación (servicios de localización, acceso a Internet, etc.)
- **Optimizado para baja potencia y poca memoria.** Por ejemplo, Android utiliza la Máquina Virtual Dalvik. Se trata de una implementación de Google de la máquina virtual de Java optimizada para dispositivos móviles.
- **Alta calidad de gráficos y sonido.** Gráficos vectoriales suavizados, animaciones inspiradas en Flash, gráficos en 3 dimensiones basados en OpenGL. Incorpora codecs estándar más comunes de audio y vídeo, incluyendo H.264 (AVC), MP3, AAC, etc.

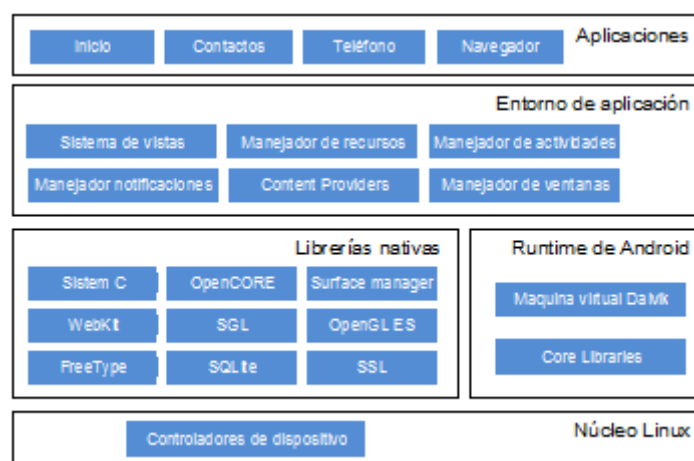


Fue desarrollado por Android Inc., empresa que en 2005 fue comprada por Google, aunque no fue hasta 2008 cuando se popularizó, gracias a la unión al proyecto de Open Handset Alliance, un consorcio formado por 48 empresas de desarrollo hardware, software y telecomunicaciones, que decidieron promocionar el software libre. Pero ha sido Google quien ha publicado la mayor parte del código fuente del sistema operativo, gracias al software Apache, que es una fundación que da soporte a proyectos software de código abierto.

▪ **Arquitectura**

Dado que Android está basado en el núcleo de Linux, tiene acceso a sus recursos, pudiendo gestionarlo, gracias a que se encuentra en una capa por encima del núcleo, accediendo así a recursos como los controladores de pantalla, cámara, memoria flash, etc.

En la figura 5.11 se distinguen claramente cada una de las capas: la que forma parte del propio núcleo de Linux, donde Android puede acceder a diferentes controladores, las librerías creadas para el desarrollo de aplicaciones Android, la siguiente capa que organiza los diferentes administradores de recursos, y por último, la capa de las aplicaciones a las que tiene acceso.



**Figura 5.11-** Arquitectura de Android [11].

## ▪ **Versiones**

Este sistema operativo, al igual que los propios teléfonos móviles, ha evolucionado rápidamente, acumulando una gran cantidad de versiones:

### ▪ **Cupcake: Android Versión 1.5**

Características: Widgets, teclado QWERTY virtual, copy & paste, captura de vídeos y poder subirlos a Youtube directamente.

### ▪ **Donut: Android Versión 1.6**

Características: Añade a la anterior la mejora de la interfaz de la cámara, búsqueda por voz, y navegación en Google Maps.

### ▪ **Eclair: Android Versión 2.0/2.1**

Características: Mejoras en Google Maps, salvapantallas animado, incluye zoom digital para la cámara, y un nuevo navegador de Internet.

### ▪ **Froyo: Android Versión 2.2**

Características: Incluye hotspot Wifi, mejora de la memoria, más veloz, Microsoft Exchange y video-llamada.

### ▪ **Ginger Bread: Android Versión 2.3**

Características: Mejoras del consumo de batería, el soporte de vídeo online y el teclado virtual, e incluye soporte para pagos mediante NFC2.

### ▪ **Honey Comb: Android Versión 3.0/3.4**

Características: Mejoras para tablets, soporte Flash y Divx, integra Dolphin, multitarea pudiendo cambiar de aplicación dejando las demás en espera en una columna, widgets y homepage personalizable.

### ▪ **Ice Cream Sandwich: Android Versión 4.0/4.1/4.3**

Características: Multiplataforma (tablets, teléfonos móviles y netbooks), barras de estado, pantalla principal con soporte para 3D, widgets redimensionables, soporte usb para teclados, reconocimiento facial y controles para PS3.

### ▪ **KitKat :Android Versión 4.4**

Características: *Google Now* es esencialmente la interfaz de Android 4.4. Es un asistente que escucha lo que preguntes y encontrará lo que estás buscando.

- **Entorno de desarrollo**

Google ha preparado el paquete de software Android SDK [17], que incorpora todas las herramientas necesarias para el desarrollo de aplicaciones en Android. En él se incluye conversor de código, depurador, librerías, emulador, documentación, ejemplos de código, etc. Todas estas herramientas son accesibles desde la línea de comandos, por otra parte para el desarrollo.

No obstante la mayoría de desarrolladores prefieren utilizar un IDE, o entorno de desarrollo integrado que integre un editor de texto con todas las herramientas de desarrollo. Aunque no son las únicas dos posibilidades las alternativas más recomendables son Eclipse e IntelliJ Idea.

### **5.2.1.5 Eclipse**

Eclipse [12] es un entorno de programación compuesto por un conjunto de herramientas de programación de código abierto multiplataforma para desarrollar lo que el proyecto llama "Aplicaciones de Cliente Enriquecido", opuesto a las aplicaciones "Cliente-liviano" basadas en navegadores. Esta plataforma, típicamente ha sido usada para entornos de desarrollo integrados (del inglés IDE), como el IDE de Java llamado Java Development Toolkit (JDT) y el compilador (ECJ) que se entrega como parte de Eclipse (y que son usados también para desarrollar el mismo Eclipse). Sin embargo, también se puede usar para otros tipos de aplicaciones cliente, como BitTorrent o Azureus.

Eclipse es también una comunidad de usuarios, extendiendo constantemente las áreas de aplicación cubiertas. Un ejemplo es el recientemente creado Eclipse Modeling Project, cubriendo casi todas las áreas de Model Driven Engineering.

Eclipse fue desarrollado originalmente por IBM como el sucesor de su familia de herramientas para VisualAge. Eclipse es ahora desarrollado por la Fundación Eclipse, una organización independiente sin ánimo de lucro que fomenta una comunidad de código abierto y un conjunto de productos complementarios, capacidades y servicios.

Eclipse fue liberado originalmente bajo la Common Public License, pero después fue re-licenciado bajo la Eclipse Public License. La Free Software Foundation ha dicho que ambas licencias son licencias de software libre, pero son incompatibles con Licencia pública general de GNU (GNU GPL).

### 5.2.1.6 OpenCV en Android

OpenCV4Android SDK [17] permite el desarrollo de aplicaciones para Android con el uso de la biblioteca OpenCV. Se encuentra disponible como un SDK con una serie de ejemplos y documentación Javadoc para la API de Java OpenCV. También contiene archivos prediseñados APK, que se pueden ejecutar en el dispositivo al instante.

La estructura del contenido se describe en la Figura 5.12.

```
OpenCV-2.4.9-android-sdk
|  _ Apk
|  |  _ OpenCV_2.4.9_binary_pack_armv7a.apk
|  |  _ OpenCV_2.4.9_Manager_2.18_XXX.apk
|
|  _ Doc
|  _ muestras
|  _ Sdk
|  |  _ Etc
|  |  _ Java
|  |  _ Nativo
|  |  _ 3rdparty
|  |  _ Jni
|  |  _ Libs
|  |  _ Armeabi
|  |  _ Armeabi-v7a
|  |  _ X86
|
|  _ LICENCIA
|  _ README.android
```

**Figura 5.12-** Estructura OpenCV4Android SDK.

- *sdk* contiene API OpenCV y bibliotecas para Android:
- *sdk/java* contiene un proyecto de biblioteca de Android Eclipse proporcionando API OpenCV Java que se puede importar en el espacio de trabajo.
- *sdk/nativos* contiene cabeceras OpenCV C++ (en el código JNI) y bibliotecas nativas de Android para ARM-v5, ARM-v7a y arquitecturas x86.

- *sdk/etc* contiene Haar y PBP cascadas distribuidas con OpenCV.
- *apk* contiene paquetes de Android que se deben instalar en el dispositivo Android de destino para permitir acceso a la biblioteca OpenCV a través de la API de OpenCV.

En los dispositivos de producción que tienen acceso a Google Play Market (e Internet), estos paquetes se instalarán en el primer inicio de una aplicación que utiliza la API de OpenCV.

- *muestras* contiene la muestra proyectos de aplicaciones y sus paquetes precompilados (APK).
- *doc* contiene diversa documentación OpenCV en formato PDF. También está disponible en línea en <http://docs.opencv.org>

Para la instalación y configuración de OpenCV4Android SDK es necesario tener el siguiente software instalado y configurado:

- JDK
- SDK y NDK Android
- Eclipse IDE
- ADT y CDT plugins para Eclipse

#### **5.2.1.7 Requisitos de hardware y software.**

Los medios necesarios para la elaboración del trabajo requieren la utilización de hardware y software específico. Los recursos necesarios que se han utilizado para realizar este trabajo son:

- PC o portátil con 2GB de RAM y sistema operativo Linux/Windows.
- Dispositivos móviles (tableta o teléfono) basados en Android.
- Conexión a Internet.
- Eclipse y lenguaje Java.
- Java Runtime Environment.
- OpenCV.
- Microsoft Office.
- Colección de imágenes.
- Conversor de imágenes.

### 5.2.1.8 Casos de uso

Un caso de uso [18] es una descripción de los pasos o las actividades que deberán realizarse para llevar a cabo algún proceso. Los personajes o entidades que participarán en un caso de uso se denominan actores. En el contexto de ingeniería del software, un caso de uso es una secuencia de interacciones que se desarrollarán entre un sistema y sus actores en respuesta a un evento que inicia un actor principal sobre el propio sistema.

A continuación se muestra el caso de uso principal:

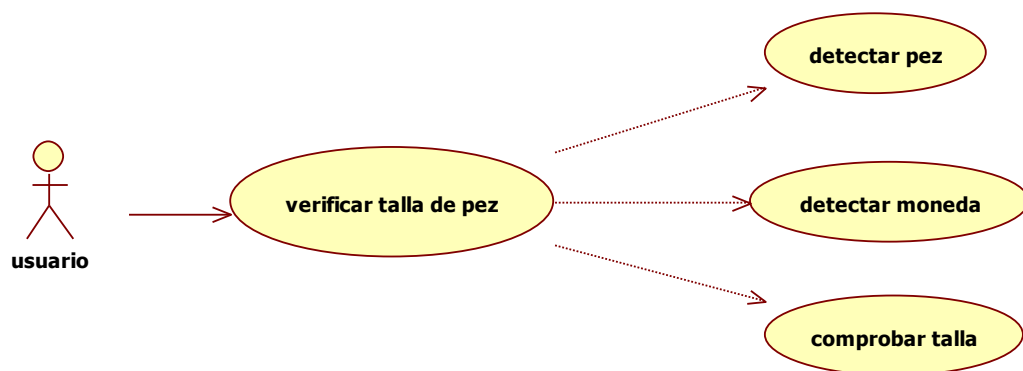


Figura 5.13- Diagrama de casos de uso.

#### ESPECIFICACIÓN DEL CASO DE USO:

##### 1. Verificar talla de pez

**Actor:** usuario.

**Precondición:** abrir la aplicación.

**Post-condición:** mensaje en pantalla indicando si el pez cumple o no la talla mínima.

**Camino normal:**

1. El sistema muestra una pantalla con un botón para comenzar.
2. El usuario pulsa este botón.
3. El sistema muestra las instrucciones del aplicativo.

4. El usuario accede a un menú donde indica el pez y el sistema carga el clasificador correspondiente.
5. El sistema activa la cámara y abre una vista nueva con el resultado de la visualización.
6. El sistema detecta el pez y la moneda.
7. El sistema accede a la base de datos y comprueba si la medida cumple la normativa.
8. El sistema muestra un mensaje indicando el resultado al usuario.

### **Camino alternativo:**

- No detecta el pez:

1. El sistema no muestra el cuadro alrededor del pez indicando su detección.
2. El sistema continúa analizando imágenes hasta detectarlo.

- No detecta el euro:

1. El sistema muestra un mensaje al usuario indicándole que no lo está detectando.
2. El sistema continúa analizando imágenes hasta detectarlo.

### **Excepciones:**

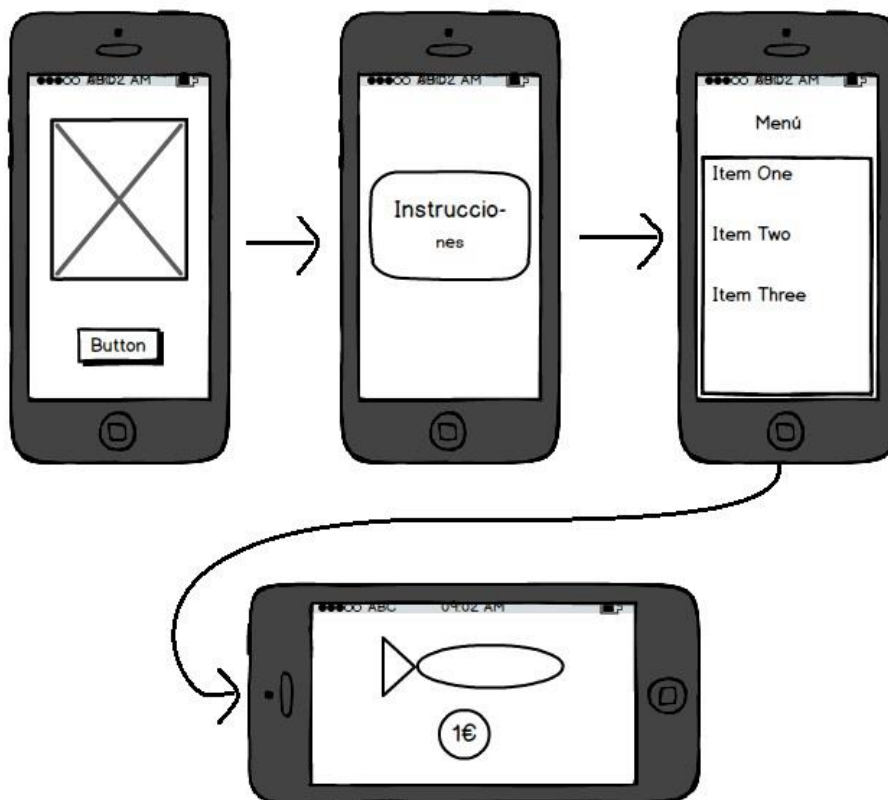
1. El dispositivo móvil no disponga de la batería suficiente para utilizar la cámara.

## 5.2.2 Diseño

En este apartado definiremos el diseño de nuestro prototipo de aplicación generando un boceto del mismo y utilizaremos el diagrama de flujos para mostrar secuencia de funcionamiento del mismo.

### 5.2.2.1 Interfaz gráfica.

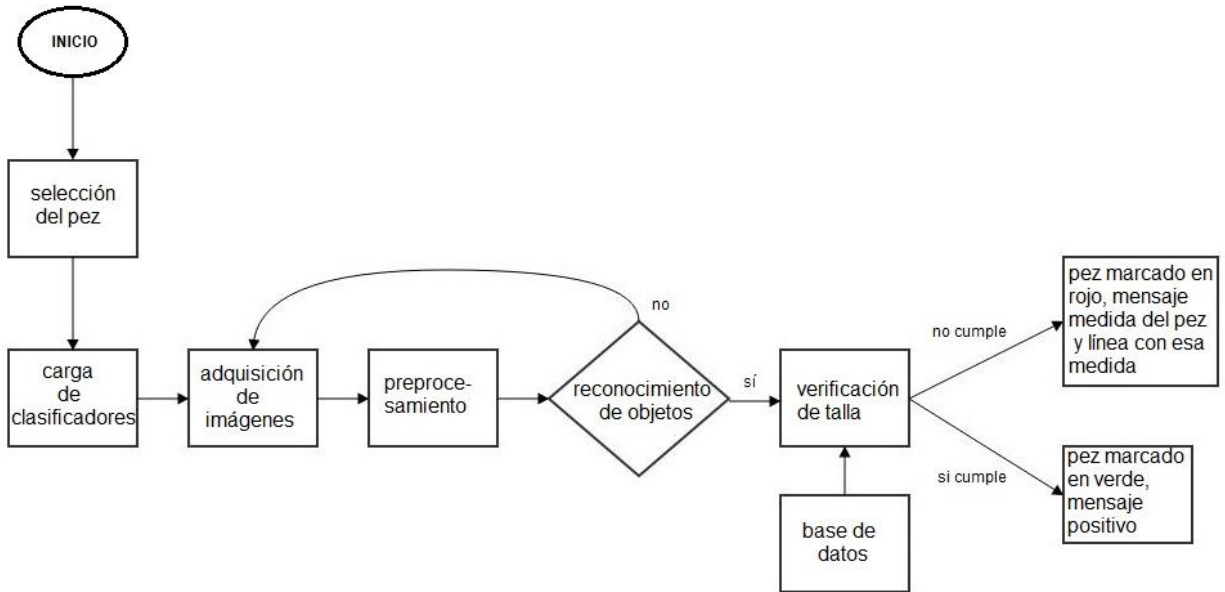
Una de las fases iniciales del desarrollo del prototipo es la realización de un boceto de la aplicación para poder tener una idea de cómo enfrentarse al diseño real de la misma. Se ha intentado ser lo más fiel posible a este primer diseño durante la implementación del mismo.





### 5.2.2.2. Diagrama de flujo

El diagrama de flujo nos permite ver los estados por los que pasa nuestra aplicación:



El funcionamiento del aplicativo sigue la siguiente secuencia:

1. Se inicia la aplicación, se le muestra al usuario las instrucciones y éste selecciona el tipo de pez.
2. La app carga el clasificador de la moneda de un euro, y el correspondiente a la especie de pez seleccionada.
3. El sistema adquiere una imagen.
4. Aplica los algoritmos de preprocesamiento para la detección.
5. Si se detectan el pez y la moneda se procede a la verificación de talla del mismo, en caso contrario el aplicativo continuará el proceso de detección, mostrando un mensaje al usuario en caso de no encontrar la moneda de un euro.
6. Detectados ambos objetos se procede a hacer un cálculo estimado de la medida del pez que se muestra en la imagen usando como referencia la medida real de la moneda y la medida en píxeles. Obtenido este dato, el sistema accede a la base de datos y comprueba la medida que debe cumplir el pez en cuestión.

7. Una vez comprobado pueden darse dos casos, que éste cumpla dicha talla o por el contrario no la cumpla, en ambos caso se le indicará al usuario:
  - No cumple: se le muestra el pez seleccionado en color rojo y sobre éste una línea con el tamaño que debería tener en pixeles. Asimismo se mostrará un mensaje de texto indicando que es demasiado pequeño.
  - Si cumple: se le muestra el pez seleccionado en color verde y un mensaje de texto informando que si cumple la medida establecida.

## **5.2.3 Implementación**

### **5.2.3.1 Entrenamiento de los detectores.**

Nuestro aplicativo requiere de archivos xml propios, tanto para los peces seleccionados, como para la moneda de un euro.

Las especies de peces seleccionadas para la creación de este prototipo han sido la sardina, la caballa y la dorada. Se han seleccionado éstas 3 especies por ser capturas habituales de nuestros mares.

En nuestro entrenamiento guiado necesitaremos un número de imágenes apropiado para poder conseguir un clasificador adecuado y los resultados sean los esperados.

El número de imágenes que necesitaremos vendrá estrechamente relacionado con la aplicación que queremos darle, si los objetos que debemos detectar representan una gran variación y el entorno puede variar considerablemente entonces necesitaremos un número de imágenes mayor.

En nuestro caso usaremos alrededor de 1000 imágenes positivas y 2500 negativas. Si los resultados no fueran los adecuados variaríamos estos valores. Es decir, si obtenemos un clasificador que no detecta bien el objeto, necesitaremos más

muestras positivas, en cambio, si el número de falsos positivos es muy elevado, será el número de imágenes negativas lo que debemos variar.

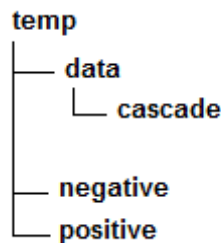
Para obtener las imágenes que necesitaremos para el aprendizaje hemos creado una colección propia de las mismas, las positivas a través de fotografiar al objeto, y las negativas de bases de datos personales en las que no aparecía el objeto en cuestión.

## Proceso

El proceso completo lleva una serie de 4 pasos:

1. Adquisición de imágenes
2. Creación de la muestra
3. Entrenamiento
4. Pruebas

Lo primero que debemos hacer es generar una estructura de directorio para poder trabajar:



Para la obtención del clasificador utilizamos el método de Viola-Jones, que para un objeto en concreto requiere el seguimiento estricto de unos pasos que se van a describir a continuación.

En primer lugar, se requerirá una preparación exhaustiva de las imágenes que servirán como muestras, tanto positivas como negativas, para el aprendizaje del clasificador. Así el clasificador obtenido tendrá una probabilidad de detección máxima y la probabilidad de falsa alarma mínima. Sin embargo, por muy elevado que sea el conjunto de muestras facilitado al algoritmo, nunca representará la totalidad del espacio muestral; motivo por el cual resulta fundamental proporcionar muestras bien seleccionadas, que representen de la mejor forma posible al total.

El primer paso del proceso es la recopilación de las imágenes, que como se explicaba anteriormente se realizó con fotografías y bases de datos propias.

El segundo paso es la preparación de las imágenes de muestra. Será necesario distinguir en esta parte, la preparación de las muestras positivas y de las negativas, ya que en el primer caso, el proceso se ha de centrar en la calidad de las mismas y en el segundo en la variedad.

Seguidamente, se pasará a la parte de entrenamiento del clasificador. Esta parte del proceso es la realizada por el programa llamado Haartraining, obteniendo al finalizar un archivo \*.xml

### **Preparación de las imágenes de muestra**

La preparación de las imágenes para entrenamiento consiste básicamente en proporcionar al Haartraining dos ficheros; el primero de ellos contendrá una lista de imágenes en las que no se encuentre el objeto a detectar, que constituirán las muestras negativas.

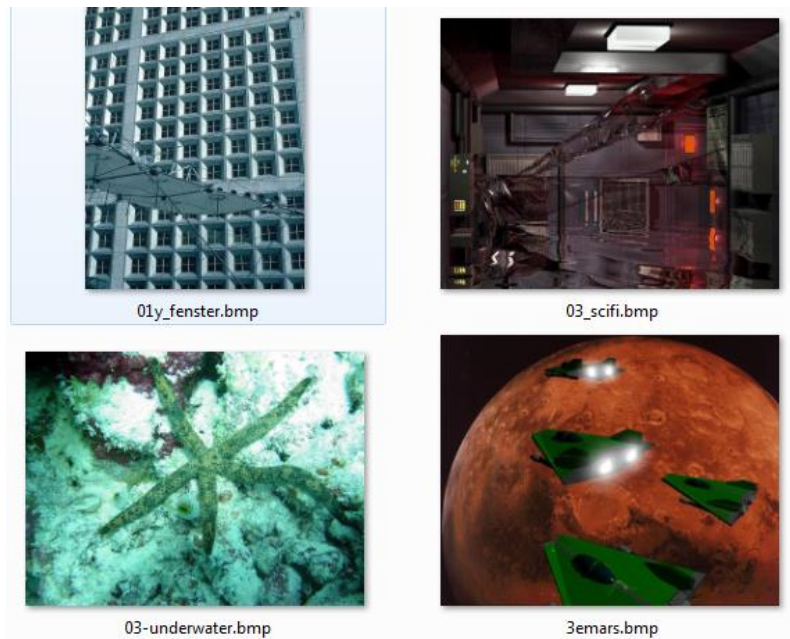
El segundo fichero proporcionado al Haartraining para el entrenamiento será el que contendrá imágenes en las que si se encuentra el objeto a detectar, formando las muestras positivas.

A continuación explicaremos la preparación de las imágenes que formarán las dos listas para ambos ficheros explicando su elaboración.

### **Muestras negativas**

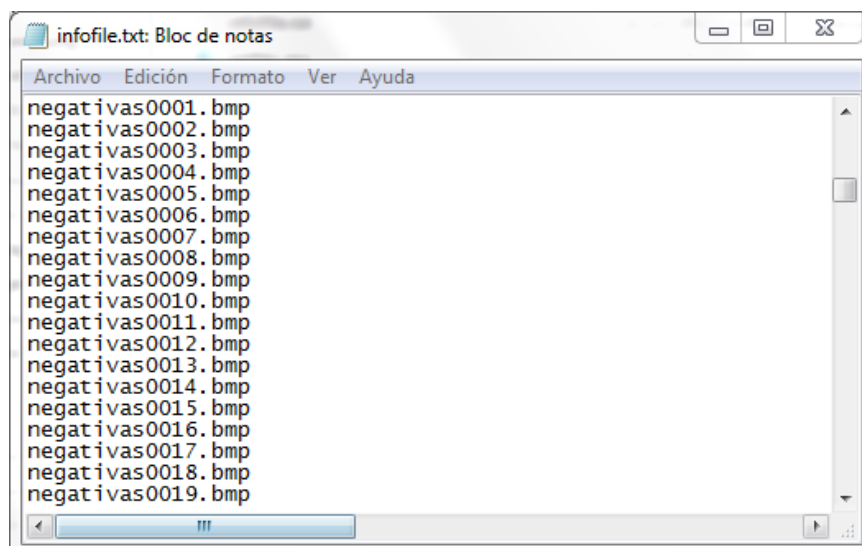
Las imágenes negativas son aquellas tomadas de forma arbitraria. Lo más importante de estas imágenes, es que no contengan el objeto a clasificar.

En la Figura 5.14 podemos ver una pequeña muestras de las utilizadas en este proceso.



*Figura. 5.14- Ejemplo de imágenes negativas.*

Las muestras negativas requieren un fichero de índices de extensión txt para que el programa pueda trabajar con ellas. Este fichero contiene el nombre de las imágenes usadas como fondos con su dirección relativa dentro del sistema de ficheros.

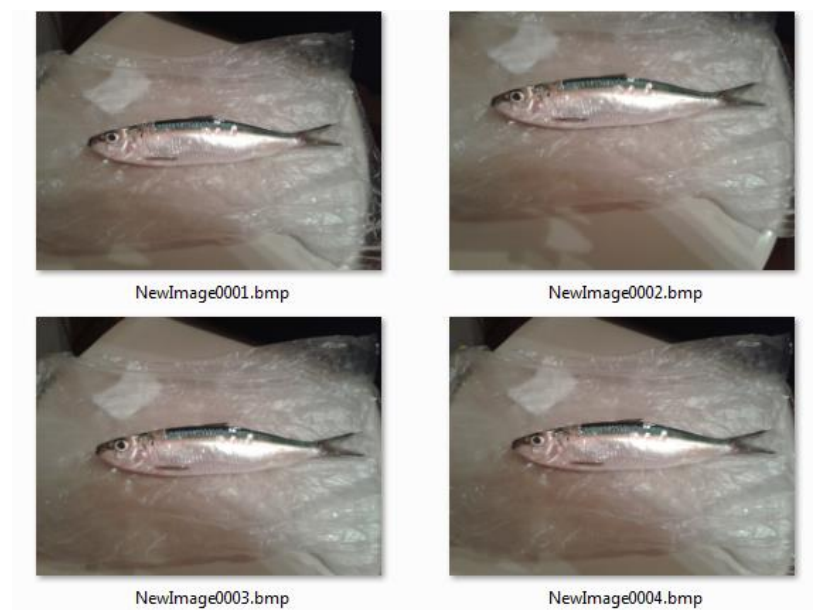


*Figura 5.15- Fichero de índices de las imágenes negativas.*

El contenido de esas imágenes debe estar suficientemente cuidado como para que sea factible sacar de ellas información para el aprendizaje.

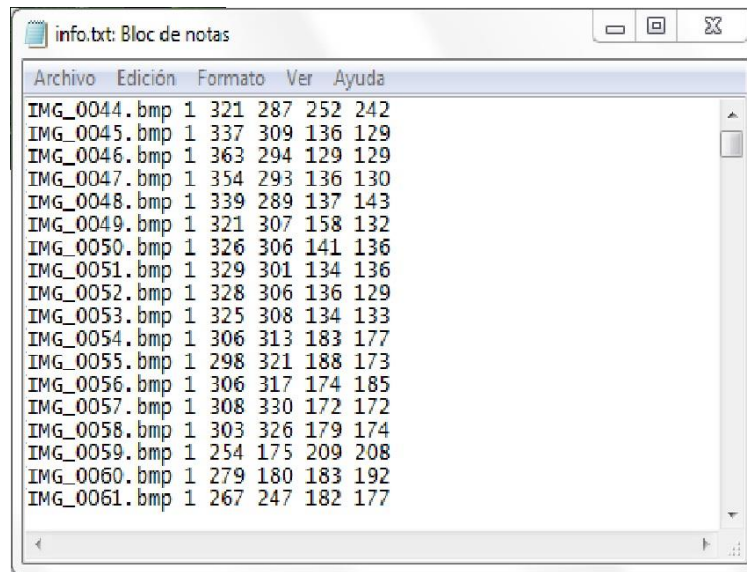
### **Muestras positivas**

Son un conjunto de imágenes en las que se encuentra el objeto a clasificar de una forma clara. Las imágenes positivas utilizadas para el desarrollo de este proyecto han sido tomadas desde diferentes ángulos, con diferentes fondos e iluminación. En la Figura 5.16 se puede observar una pequeña muestra de estas imágenes:



**Figura 5.16-** Ejemplo de muestras positivas.

Este fichero de muestras positivas, debe ir acompañado de un fichero en que se indicará el listado de imágenes acompañadas además del número de muestras positivas (objetos a detectar) que contiene dicha imagen y de las coordenadas en las que se encuentran las muestras dentro de ella en formato  $x1\ y1\ \Delta x\ \Delta y$ , tantas veces como objetos indique el número antes indicado, donde  $(x1,y1)$  será el punto correspondiente a la esquina superior izquierda de la muestra,  $\Delta x$  y  $\Delta y$  será respectivamente el ancho y el alto de la muestra en esa imagen, partiendo desde el punto anterior.



**Figura 5.17-** Fichero de índice de las imágenes positivas.

Para obtener este fichero se utiliza el programa Objectmarker [12], el cual toma como entrada un fichero que contenga una lista de imágenes a visualizar (las imágenes deben estar en formato .bmp), lo abre y va leyendo línea a línea.

Representa por pantalla la imagen referenciada por cada línea del fichero. Una vez representada la primera de las imágenes, el programa permanece en estado de espera hasta la llegada de algún evento de teclado o de ratón para los que está preparado para responder.

Los eventos de teclado y las acciones que provocan en el programa son los siguientes:

- <Enter>: Guarda los rectángulos añadidos y muestra la siguiente imagen
- <ESC>: Sale del programa
- <Espacio>: Añade las coordenadas del rectángulo en la imagen actual

Para una determinada imagen, se hace “click” con el ratón sobre un determinado punto y se arrastra, dibujando un rectángulo con el que podremos englobar el objeto que deseemos.

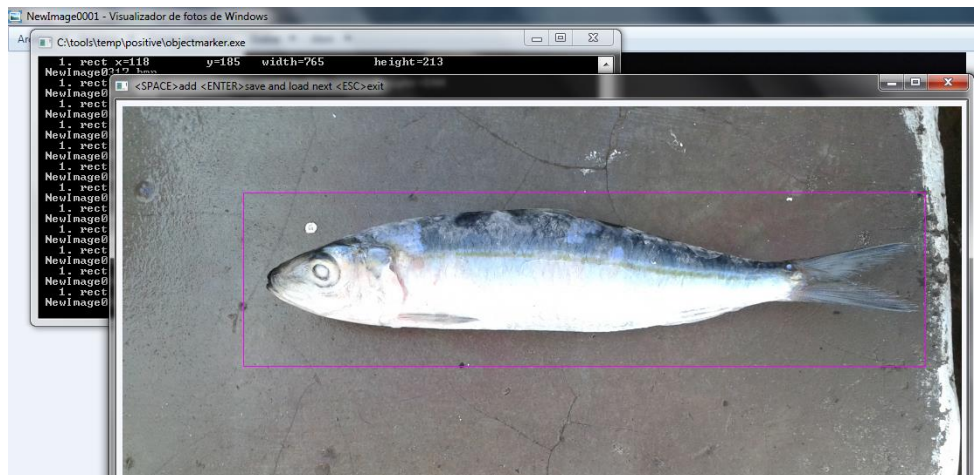


Figura 5.18- Ejemplo de uso Objectmarker.

### Crear muestra

Una vez adquiridas las imágenes positivas y negativas y generados los archivos correspondientes, se pasará a la siguiente etapa, crear el conjunto de muestras que extrae de ambos conjuntos de imágenes las zonas adecuadas. Para ello se utilizará el programa Createsamples.

Tras realizar este proceso se creará un vector que contiene las imágenes de muestra. La llamada al programa desde el intérprete de comandos para crear el fichero .vec ha sido la siguiente:

```
opencv_createsamples.exe - info positive/info.txt - vec data/vector.vec - num 1000 -w 24 - h 24
```

- info: Precederá al nombre del fichero lista que contendrá todas las imágenes de los objetos que constituyen las muestras positivas y cuya construcción se ha descrito en la sección anterior.
- vec: Tras esta etiqueta escribiremos el nombre del fichero .vec que creará el Createsamples a partir de las muestras proporcionadas.
- num: Indica al Createsamples el número de muestras que contiene el fichero lista y que por tanto añadirá en el formato adecuado en el fichero .vec.



- *w*: Indica el ancho deseado para las muestras creadas en la ejecución del *Createsamples*. Su valor por defecto es de 24 píxeles.
- *h*: Indica el valor para el alto de las muestras. Su valor por defecto es de 24 píxeles.

Una vez preparadas las imágenes, ya se puede iniciar el entrenamiento de un clasificador mediante el método de entrenamiento.

### **Entrenamiento:**

Para esta etapa nos valdremos de otra herramienta llamada Haartraining, el archivo índice generado con las imágenes negativas y el archivo de muestras generado en la etapa anterior.

Mediante la siguiente línea de comando ejecutamos el programa.

```
opencv_haartraining.exe -data data/cascade -vec data/vector.vec -bg
negative/infofile.txt -npos 1000 -nneg 2525 -nstages 30 -mem 500 -mode all -w
24 -h 24 -nsplits 2 -minhitrate 0,999
```

El significado de cada uno de los argumentos:

- *data*: Se utiliza para especificar el directorio donde el programa irá almacenando el clasificador a medida que lo construya. Haartraining crea un directorio para cada una de las etapas con el nombre del número de la etapa, donde almacena el archivo *AdaBoostCARTHaarClassifier.txt* cuya estructura hemos visto en el apartado anterior.
- *vec*: Tras esta etiqueta se especifica el fichero con extensión *.vec* creado mediante *Createsamples*, que contiene la información de las muestras positivas.
- *bg*: Precede al fichero que contenga la lista de imágenes de fondo en las que se buscarán los candidatos negativos.
- *npos*: Se introducen tras esta etiqueta el número de muestras positivas que contiene el fichero *.vec*. Su valor por defecto es 2000.

- `nneg`: Aquí se introduce el número de muestras negativas que se le solicita al entrenamiento que encuentre en las imágenes de fondo del tamaño especificado. Su valor por defecto también es 2000.
- `nstages`: Sirve para especificar el número de etapas deseadas de las que se compondrá el clasificador. El valor por defecto es 14.
- `mem`: Selecciona la memoria que se va a utilizar para el proceso de entrenamiento.
- `mode`: Permite escoger para las características, si se desea el conjunto básico (BASIC) o el conjunto extendido (ALL). La forma por defecto es la primera.
- `w`: Sirve para especificar el ancho de las muestras positivas proporcionadas. Su valor por defecto es 24 píxeles.
- `h`: Sirve para especificar el alto de las muestras positivas. Su valor por defecto es también 24 píxeles.
- `nsplits`: Hace referencia a la estructura del árbol de clasificación, que determina la debilidad del clasificador. En nuestro caso se utilizará el clasificador con 2 divisiones internas de nodos.
- `minhitrate`: Mínima tasa de detección que se desea alcanzar en cada etapa, siendo la tasa del clasificador completo ese mismo valor elevado al número de etapas. El valor por defecto es 0,995.

Durante todo el transcurso del proyecto se estuvieron realizando entrenamientos de diferentes clasificadores para cada una de las especie de peces utilizadas, y para la moneda de un euro.

El comando `haartraining` genera un archivo `xml` cuando el proceso haya terminado. Si se desea convertir los archivos de salida del comando `haartraining` en un archivo `xml`, si éste aún no ha concluido con el entrenamiento, se debe usar la aplicación `cascade2xml`.

La llamada a la aplicación cascade2xml para nuestro detector sería la siguiente:

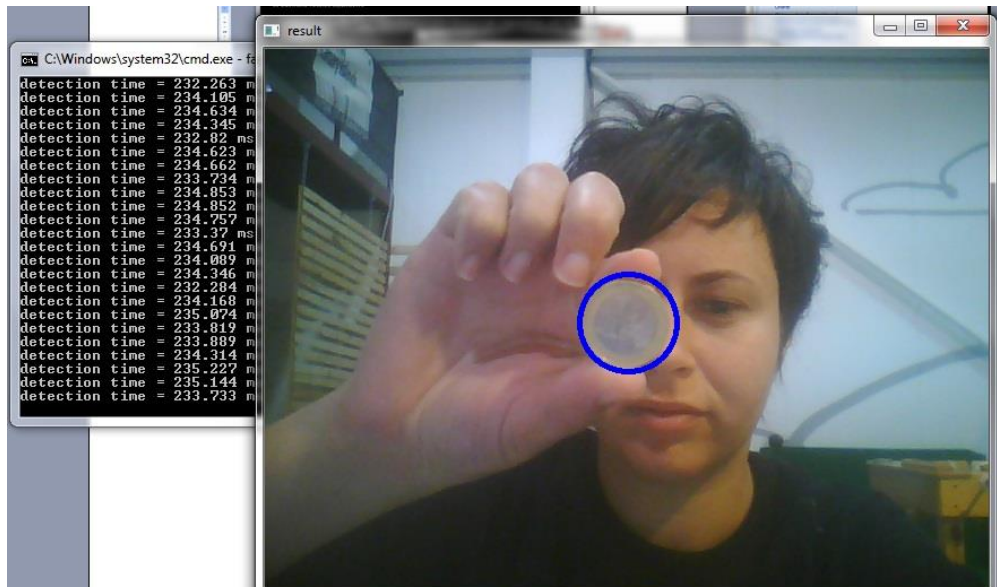
```
./cascade2xml.exe data cascada.xml 24 24
```

Donde,

- data: Indica el directorio donde se han almacenado los archivos del clasificador.
- cascada.xml: Se especifica el archivo con extensión .xml creado por el cascade2xml, que contiene la información de los archivos del clasificador.
- 24 24: Es el ancho y alto, respectivamente, de las muestras utilizadas durante el entrenamiento con el Haartraining.

### Pruebas:

Utilizando nuestro archivo xml del ejemplo que nos trae la librería OpenCV podemos comprobar el funcionamiento de nuestro propio clasificador.



*Figura. 5.19- Resultado de detección de moneda de 1 euro.*

Como se observa en la Figura 5.19, el resultado de la detección, aplicando el detector de monedas de 1 euro, se resalta en azul.

### 5.2.3.2. Desarrollo del software prototipo de detección.

Una vez generados los clasificadores necesarios para el aplicativo, hemos procedido a la codificación del mismo. Para ello ha sido necesaria la instalación previa del entorno de trabajo y la integración del OpenCV 4 SDK en el mismo.

- **Interfaz de usuario:**

La aplicación está formada por un conjunto de pantallas que permiten la interacción. Hemos intentando en todo momento respetar el diseño planteado inicialmente.

Estas pantallas reciben el nombre de actividades y son objetos que extienden de la clase Activity.

Cada una de sus actividades y sus correspondientes elementos se encuentran en ficheros XML independientes del código.

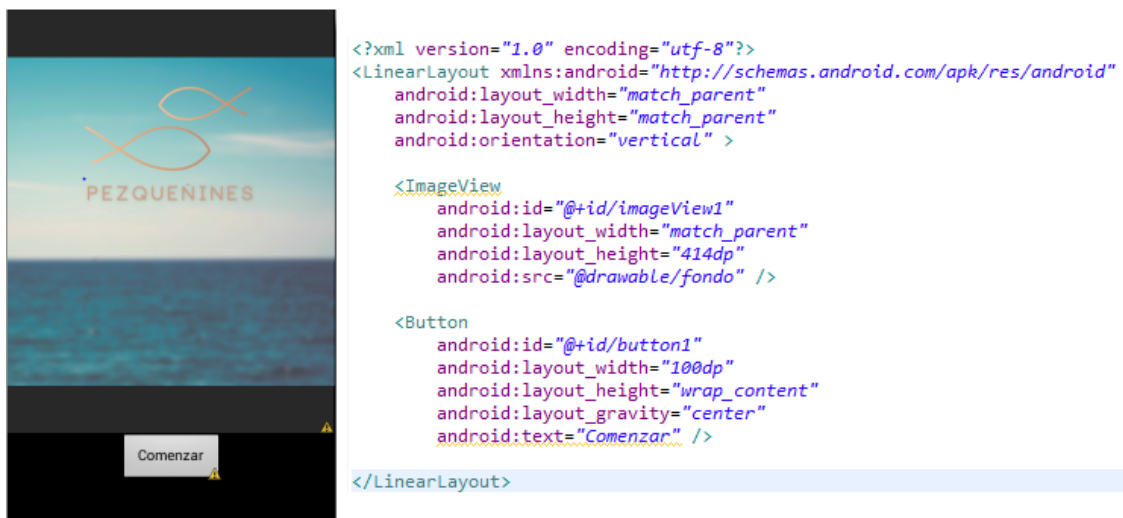


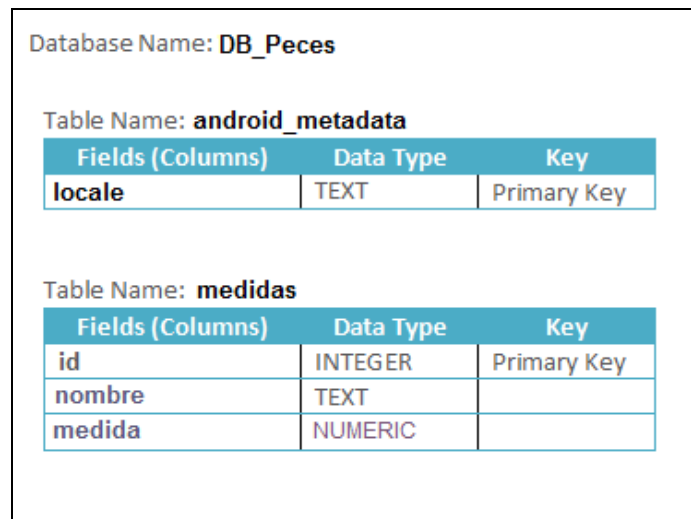
Figura 5.20- Actividad inicial y su correspondiente fichero XML.

- **Base de datos:**

Para almacenar las tallas mínimas reglamentarias hemos creado una base de datos SQLite, ya que Android incorpora de serie todas las herramientas necesarias para la creación y gestión de las mismas.

Entre ellas cuenta con una completa API para llevar a cabo de manera sencilla todas las tareas necesarias. A través de una clase auxiliar llamada SQLiteOpenHelper, o para ser más exactos, de una clase propia que deriva de ella, incorporamos nuestra base de datos a la aplicación y conectamos con ella para realizar las tareas necesarias.

Nuestra base datos de Pezqueñines está compuesta por dos tablas, las cuales se estructuran como se muestra en la Figura 5.21.



Database Name: **DB\_Peces**

Table Name: **android\_metadata**

Fields (Columns)	Data Type	Key
<b>locale</b>	TEXT	Primary Key

Table Name: **medidas**




Fields (Columns)	Data Type	Key
<b>id</b>	INTEGER	Primary Key
<b>nombre</b>	TEXT	
<b>medida</b>	NUMERIC	

*Figura 5.21- Estructura de la base de datos DB\_Peces.*

La primera de ellas es una tabla con el nombre **android\_metadata**, con una única columna de nombre **locale** y con una fila con el valor en\_ES.

La segunda es una tabla llamada **medida**, que posee tres columnas. La primera de ellas contiene un identificador único de cada elemento, y las posteriores contienen el nombre y la medida en centímetros de las especies utilizadas.

Como nombramos anteriormente hemos utilizado para el prototipo las especies sardina, dorada y caballa con sus respectivas tallas mínimas establecidas por el Real Decreto 560/1995, de 7 de abril [13], las cuales podemos observar en la Figura 5.22.

Id	nombre	Medida (cm)
1	Sardina 	11
2	Dorada 	19
3	Caballa 	20

*Figura 5.22-Tabla de medidas mínimas.*

En caso de que fuese necesario actualizar el contenido de nuestra base de datos usaríamos el método *onUpdate* de la clase *SQLiteOpenHelper*, indicándole la tabla y los valores de los registros que deseamos que sean actualizados.

- **Funcionalidad:**

A la interfaz de usuario desarrollada previamente le añadimos la funcionalidad de detección, con las correspondientes modificaciones necesarias para que la detección se adapte al objetivo de nuestro prototipo.

Como pudimos observar en el diagrama de flujos, lo primero que realizará será la carga de los clasificadores necesarios, en este caso cargamos los clasificadores generados previamente.

```

//-----load moneda classifier -----

InputStream is = getResources().openRawResource(
    R.raw.moneda);
File cascadeDir = getDir("cascade", Context.MODE_PRIVATE);
mCascadeFile = new File(cascadeDir, "moneda.xml");
FileOutputStream os = new FileOutputStream(mCascadeFile);

byte[] buffer = new byte[4096];
int bytesRead;
while ((bytesRead = is.read(buffer)) != -1) {
    os.write(buffer, 0, bytesRead);
}
is.close();
os.close();

```

**Figura 5.23** – Código ejemplo de carga de clasificador.

Una vez cargados, el sistema procederá a la detección de los objetos, siguiendo el algoritmo usado para la detección de objetos OpenCV, pero a diferencia de éste, nuestro detector debe realizar el proceso para la detección tanto de la moneda de euro como para el pez simultáneamente.

```

MatOfRect pez = new MatOfRect();
MatOfRect moneda = new MatOfRect();

if (mDetectorType == JAVA_DETECTOR) {
    if (mJavaDetector != null)
        mJavaDetector.detectMultiScale(mGray, moneda, 1.1, 2,
            2, // TODO: objdetect.CV_HAAR_SCALE_IMAGE
            new Size(mAbsoluteFaceSize, mAbsoluteFaceSize),
            new Size());

    if (mJavaDetector2 != null)
        mJavaDetector2.detectMultiScale(mGray, pez, 1.1, 2,
            2, // TODO: objdetect.CV_HAAR_SCALE_IMAGE
            new Size(mAbsoluteFaceSize, mAbsoluteFaceSize),
            new Size());
}

```

**Figura 5.24** – Código ejemplo de detección.

Detectados ambos objetos se procede a la verificación de la talla como vimos anteriormente en la secuencia de funcionalidad que sigue la app, mostrando en cada caso el resultado correspondiente al usuario final.

Esta verificación de la talla se realiza utilizando como referencia la medida real de la moneda en cm, y las medidas estimadas en píxeles tanto de la moneda como del

pez, con las cuales podemos hallar la medida aproximada del pez en cm haciendo una regla de tres:

$$\text{resultado} = (\text{ancho\_rectangulo\_pez} * \text{anchomoneda}) / \text{ancho\_rectangulo\_moneda}$$

Veamos a continuación un ejemplo al realizar la captura de una sardina, si en el momento de la detección el ancho en píxeles del rectángulo que rodea la moneda es de 118 píxeles y el ancho del rectángulo que pez es de 461 píxeles, conociendo la medida real de la moneda de un euro que es 2.325 cm, aplicamos la regla anterior y el siguiente resultado sería el siguiente:

$$9.08 = (461 * 2.325) / 118$$

Obtenida dicha medida, se procede a comprobar si este dato es mayor al almacenado en la base de datos como medida establecida como talla mínima para el consumo de dicha especie.

En este caso para una sardina la medida establecida es de 11 cm por lo que este ejemplar no cumpliría con el criterio de medida mínima al medir 9.08 cm.

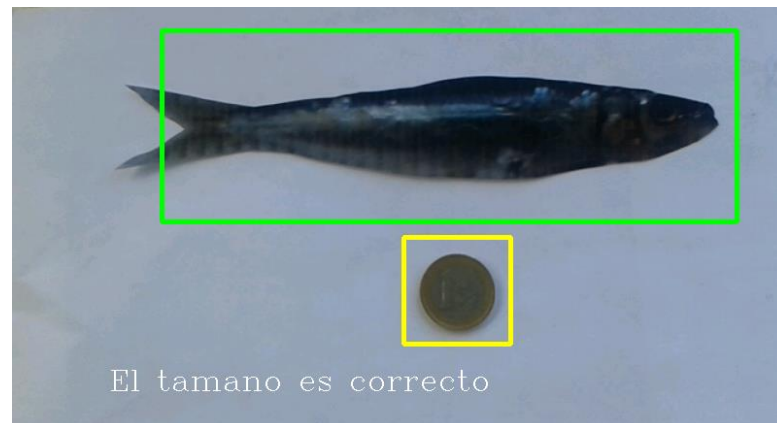
En la Figura 5.25 podemos observar como el sistema muestra al usuario que dicho ejemplar no cumple con la talla mínima, indicándoselo a través de un mensaje rodeando al pez con un rectángulo en color rojo.



**Figura 5.25** –Resultado de pez que no cumple la talla mínima.



Si por el contrario, dicho ejemplar hubiese tenido una medida superior a la establecida para su consumo mínimo el sistema mostrará al usuario el pez rodeado por un rectángulo en verde y un mensaje indicándole que es correcto, como se muestra en la Figura 5.26.



**Figura 5.26** –Resultado de pez que cumple la talla mínima.

Será requisito indispensable que el usuario coloque la moneda de un euro junto al pez para poder realizar la verificación, por el ello el sistema indicará al mismo que coloque un euro si no lo detecta.



**Figura 5.27** –Mensaje del sistema en caso de no detectar el euro.

## 6. CONCLUSIONES

El objetivo principal de este proyecto era el desarrollo de una aplicación para dispositivos móviles, que permita determinar si un pez alcanza la talla mínima para el consumo. Dicho objetivo ha sido cumplido con la implementación del prototipo en la plataforma de desarrollo Android.

Para lograr dicho objetivo era necesario cumplir unos hitos fundamentales y necesarios para el óptimo desarrollo del mismo:

- El primero de ellos era entender las bases teóricas que estaban detrás del proyecto que quería desarrollar, para poder utilizar una implementación del algoritmo de Viola-Jones y la librería de visión OpenCV. El estudio teórico me resultó un poco complicado al principio, por tratarse de conceptos totalmente nuevos para mí, con los cuales no estaba acostumbrada a trabajar.
- Posteriormente debía realizar el entrenamiento de detectores propios, éste sin duda fue uno de los hitos más complicados, tanto por la cantidad de subtareas que se podían derivar, como por su complejidad y tiempo que conllevan las mismas.

Para ello, en primer lugar debíamos obtener las muestras tanto positivas como negativas para nuestro entrenamiento. Las imágenes negativas no resultaron ser muy difíciles de obtener, sin embargo las imágenes positivas, al no encontrar bases de datos adecuadas, decidí obtenerlas realizando fotografías al objeto a detectar, lo cual resultó ser una tarea bastante costosa en términos temporales.

En segundo lugar, la obtención de los ficheros necesarios y el propio proceso de entrenamiento necesitaban mucho tiempo, en algunos casos días, tras los cuales el resultado del mismo no era el esperado y debía volver a repetir el proceso.

Superadas éstas etapas ya pasamos al desarrollar el prototipo de la aplicación en Android, el cual ha sido relativamente factible gracias a la gran comunidad que hay trabajando en ella, encontrando fácilmente información y tutoriales.

En general la consecución de este proyecto ha significado para mí un gran logro personal y profesional, con el que he podido adquirir nuevos conocimientos y capacidades que me serán de gran utilidad en mi futuro.

El desarrollo del prototipo en la plataforma Android me ha ayudado a adquirir mayor experiencia sobre esta plataforma, y me ha permitido familiarizarme con su entorno de desarrollo en Eclipse, utilizando el sistema de control de versiones Git[ 22], lo cual me abre puertas hacia futuros proyectos de desarrollo de aplicaciones móviles en la misma.

Por otro lado el uso de la Librería OpenCV ha conllevado la adquisición de nuevos conceptos en el área de la visión por computador desconocidos hasta el momento para mí, aportándome una base sólida de aprendizaje en este campo.

## **6.1 Trabajos futuros**

En este apartado describiremos las posibles tareas que se podrían desarrollar para mejorar nuestro proyecto y algunas otras partiendo del trabajo realizado en el mismo.

1. **Mejora de los clasificadores:** Una posible mejora sería crear nuevos clasificadores con un mayor número de imágenes y diferentes parámetros que consiguiesen mejores resultados en la detección.
2. **Reconocimiento de objetos:** Una posible mejora del prototipo sería que el usuario no tuviese que indicar que pez desea medir, sino que simplemente la app fuese capaz de reconocer dicho objeto por sus características visibles, concretamente su forma y diferenciarlo del resto.
3. **Mejora en el prototipo Android:** El desarrollo en esta plataforma, con una de las tecnologías más punteras, abre un abanico de posibilidades de

mejoras muy amplio, tanto en lo que a interfaz se refiere, como a nuevas funcionalidades.

4. **Desarrollo en la plataforma iOS:** OpenCV cuenta al igual que para Android, con una API para iOS, por lo cual podríamos desarrollar este mismo proyecto en esa plataforma.

## 7. BIBLIOGRAFÍA

- [1] [http://es.wikipedia.org/wiki/Pezque%C3%B1ines\\_%C2%A1No,\\_gracias!](http://es.wikipedia.org/wiki/Pezque%C3%B1ines_%C2%A1No,_gracias!)
  
- [2] [Ley 33/1980, de 21 de junio](#)
  
- [3] Rapid object detection using a boosted cascade of simple features  
P. Viola, M.J. Jones Computer Vision and Pattern Recognition, 2001. CVPR 2001
  
- [4] Robust real-time face detection P. Viola, M.J. Jones International journal of computer vision 57 (2), 137-154
  
- [5] <http://opencv.org/>
  
- [6] Tutorial Viola-Jones, M. Castrillón Santana, ULPGC 2009
  
- [7] <http://www.from.es/>
  
- [8] A general framework for object detection, Papageorgiou, Oren and Poggio, International Conference on Computer Vision, 1998.
  
- [9] <http://www.gartner.com/newsroom/id/1622614>
  
- [10] <http://developer.android.com/>
  
- [11] <http://www.androidcurso.com/>
  
- [12] <https://www.eclipse.org>
  
- [13] <https://github.com/yocox/object-marker>
  
- [14] [https://www.boe.es/diario\\_boe/txt.php?id=BOE-A-1995-8639](https://www.boe.es/diario_boe/txt.php?id=BOE-A-1995-8639)
  
- [15] Roger S. Pressman, "Ingeniería del Software: Un enfoque práctico", 6ª Edición, Editorial McGraw Hill, 2005

- [16] <http://masteringenieriasoft.blogspot.com.es/>
- [17] <http://developer.android.com/sdk/index.html>
- [18] [http://es.wikipedia.org/wiki/Caso\\_de\\_uso](http://es.wikipedia.org/wiki/Caso_de_uso)
- [19] <https://play.google.com/store/apps/details?id=com.andromo.dev181817.app291585>
- [20] <https://play.google.com/store/apps/details?id=com.dsbmobile.prodelphinusv2>
- [21] <http://visionlabs.cl/blog/?tag=reconocimiento-de-patrones>
- [22] <http://git-scm.com/book/es/Empezando-Fundamentos-de-Git>

## ANEXO 1: Instalación del entorno de desarrollo Android.

Una instalación con IDE Eclipse requiere la instalación de los siguientes elementos:

- Java Runtime Environment 5.0 o superior.
- Eclipse (Eclipse IDE for Java Developers).
- Android SDK (Google).
- Eclipse Plug-in (Android Development Tools - ADT).

Describiremos a continuación el proceso a seguir para instalar el *software* anterior. Si ya tenemos instalado Eclipse en nuestro ordenador podemos completar la instalación añadiendo *Android SDK* y *Eclipse Plug-in*. Se mantendrá la configuración actual y simplemente se añadirán nuevas funcionalidades.

### Instalación de la máquina virtual Java

Este *software* va a permitir ejecutar código Java en un equipo. A la máquina virtual Java también se la conoce como entorno de ejecución Java, Java Runtime Environment (JRE) o Java Virtual Machine (JVM).

Para instalar la Máquina Virtual Java podemos descargarla desde este enlace:

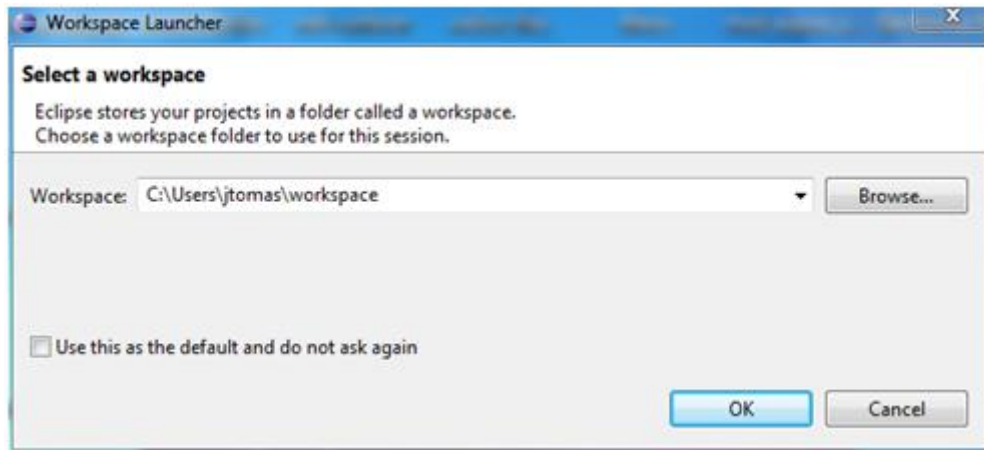
<http://www.oracle.com/technetwork/java/javase/downloads/index.html>

### Instalación de Eclipse

Para instalar Eclipse hay que seguir los siguientes pasos:

1. Acceder a la página <http://www.eclipse.org/downloads/> y descargar la última versión de "Eclipse IDE for Java Developers". Se encuentra disponible para los sistemas operativos más utilizados, como Windows, Linux y Mac OS.
2. Este software no requiere una instalación específica, simplemente descomprimir los ficheros en la carpeta que prefieras.

3. Al arrancar Eclipse comenzará preguntándonos que carpeta queremos utilizar como workspace. En esta carpeta serán almacenados los proyectos que crees en Eclipse.



*Figura anexo 1.1-Ventana inicial de Eclipse.*

4. Aparecerá una ventana de bienvenida. Cerrándola aparecerá el entorno de desarrollo.

## **Instalar Android SDK de Google**

El siguiente paso va a consistir en instalar Android SDK de Google.

1. Acceder a la siguiente página <http://developer.android.com/sdk> y descargar el fichero correspondiente a tu sistema operativo.
2. Este software no requiere una instalación específica, simplemente descomprimir los ficheros en la carpeta que se prefieran.
3. Ejecutar el programa SDK Manager.
4. Seleccionar los paquetes a instalar. Aparecerá una ventana donde podremos seleccionar los paquetes a instalar. Si se desea se puede instalar todos los paquetes (Accept All), en este caso el proceso de instalación puede tardar más de una hora. Si no dispones de tanto tiempo podemos



seleccionar solo algunos paquetes. Siempre resulta interesante instalar la última versión de Android (incluyendo documentación, ejemplos y por supuesto la plataforma). Más adelante se podría instalar más paquetes si fuese necesario.

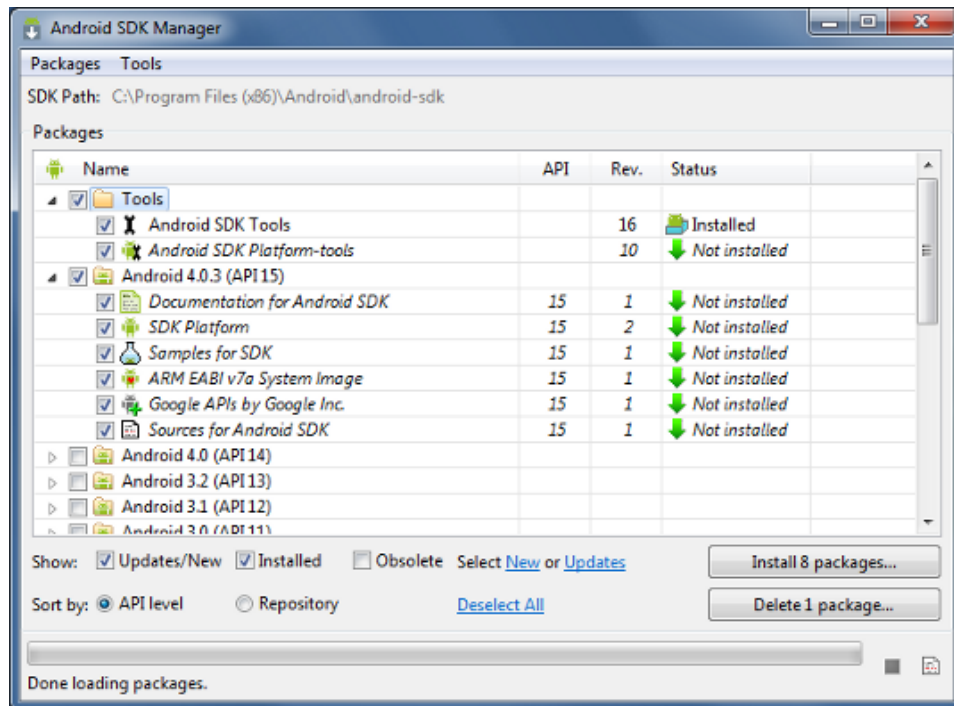


Figura anexo 1.2-Ventana para seleccionar paquetes a instalar.

## Instalación del plug-in Android para Eclipse (ADT)

El último paso consiste en instalar el plug-in Android para Eclipse, también conocido como ADT. Este software desarrollado por Google, instala una serie de complementos en Eclipse, de forma que el entorno de desarrollo se adapte al desarrollo de aplicaciones para Android. Se crearán nuevos botones, tipos de aplicación, vistas,... para integrar Eclipse con el Android SDK que acabamos de instalar.

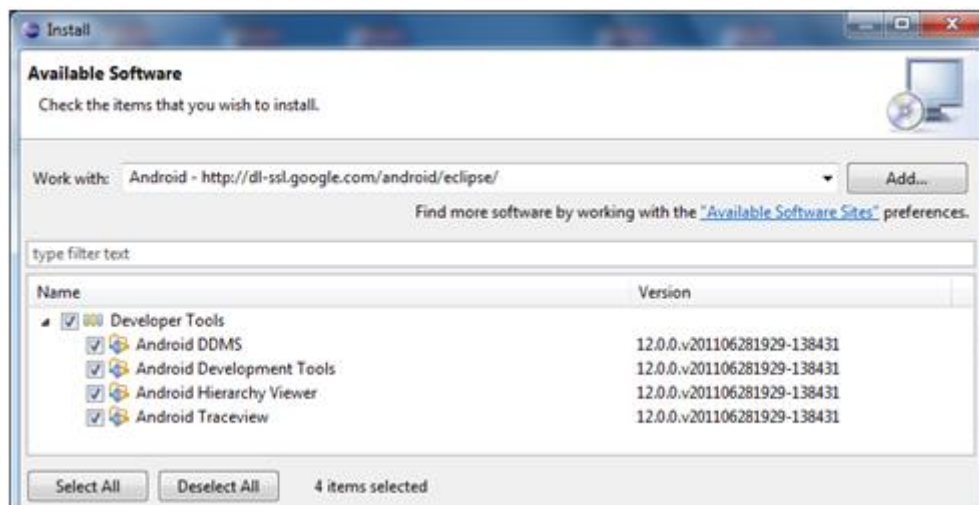
Para instalar el plug-in Android sigue los siguientes pasos:

1. Arrancar Eclipse y seleccionar *Help>Install New Software...*

2. En el diálogo Available Software que aparece, hacer clic en Add... En el cuadro de diálogo Add Site que sale introducir un nombre para el sitio remoto (por ejemplo, Plug-in Android) en el campo Name. En el campo Location, introduce la siguiente URL:

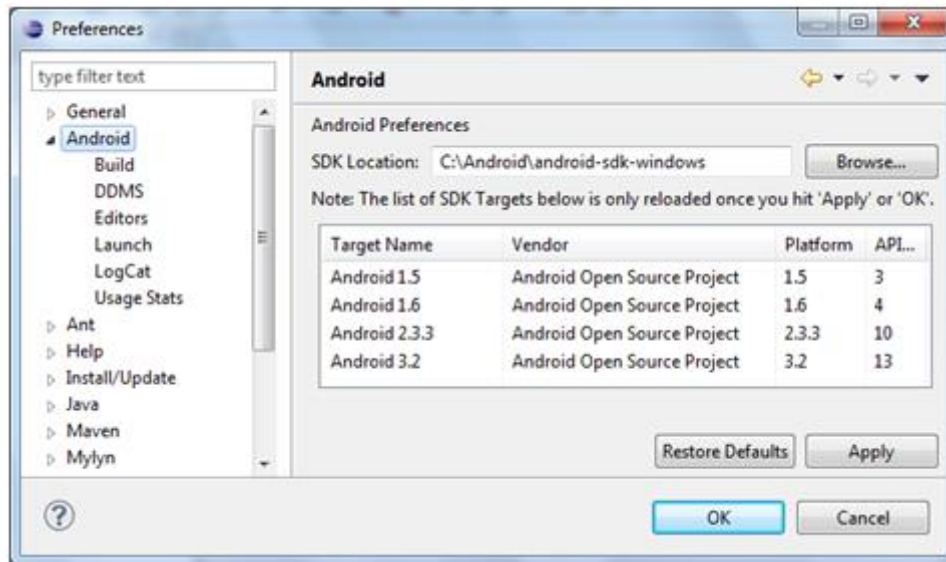
<http://dl-ssl.google.com/android/eclipse/>

Ahora en el cuadro *Available Software* debe aparecer *Developer Tools*:



*Figura anexo 1.3-Ventana de diálogo Available.*

3. Seleccionar los paquetes a instalar y pulsa Next. Ahora aparecen listadas las características de Android DDMS y Android Development Tools.
4. Pulsar Next para leer y aceptar la licencia e instalar cualquier dependencia y pulsa Finish.
5. Reiniciar Eclipse.
6. Configurar Eclipse para que sepa donde se ha instalado Android SDK. Para ello entramos las preferencias en *Windows>Preferences...* y seleccionamos Android del panel de la izquierda. Ahora pulsamos *Browse...* para seleccionar el *SDK Location* y elegimos la ruta donde en hayamos descomprimido Android SDK. Aplicamos los cambios y pulsa *OK*.



*Figura anexo 1.4-Ventana de preferencias de Eclipse.*

## ANEXO 2: Guía de Instalación de OpenCV para Windows

A continuación se presenta el proceso de instalación de los componentes necesarios para desarrollar aplicaciones de visión por computadora utilizando OpenCV en lenguaje C. Primero se muestra una manera de instalar el paquete OpenCV, este paquete es el que contiene las funciones y algoritmos de visión. Segundo, se muestra como instalar MinGW, necesario para compilar nuestro código en lenguaje C. Tercero, se muestra como instalar CodeBlocks, un entorno de desarrollo integrado que facilitará los procesos de edición de código y compilación. Cuarto, se presenta como instalar pkg-config, programa que permitirá llamar las bibliotecas necesarias para compilar el programa OpenCV. Finalmente se presenta un programa ejemplo para comprobar que todo haya sido instalado y configurado correctamente.

El paquete para la instalación de OpenCV puede ser descargado de la siguiente dirección:

<http://sourceforge.net/projects/opencvlibrary/files/opencv-win/2.2/>

### Procedimiento

Al ejecutar el archivo descargado, se abrirá el asistente de instalación de OpenCV 2.2. Se recomienda seguir los siguientes pasos:

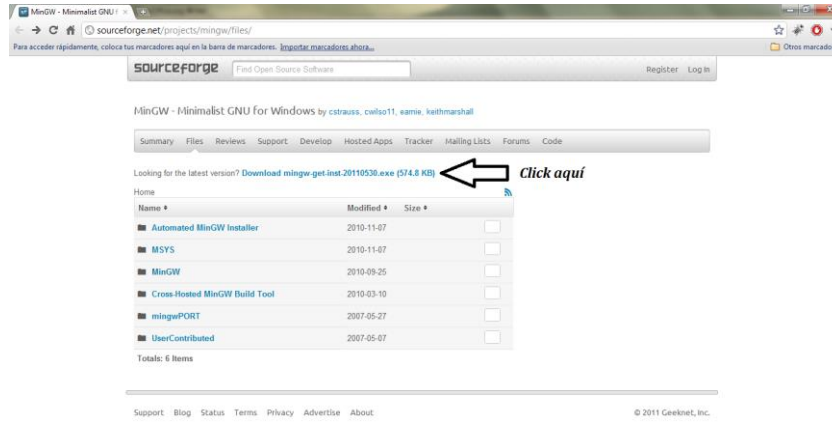
1. Primera pantalla. Dar click en “siguiente”
2. Segunda pantalla. Aceptar el contrato de licencia
3. Tercera pantalla. Seleccionar “Add opencv to the system PATH for all users”
4. Cuarta pantalla. Aceptar instalación en carpeta C:\OpenCV2.2.
5. Quinta pantalla. Dar click en “siguiente”
6. Sexta pantalla. Dar click en “Instalar”

### Instalación de MinGW

MinGW (Minimalist GNU for Windows), anteriormente conocido como MinGW32, es una implementación de los compiladores GCC para la plataforma Win32, que permite migrar la capacidad de este compilador en entornos Windows.

El paquete para la instalación de MinGW puede ser descargado de la siguiente dirección:

<http://sourceforge.net/projects/mingw/files/>



**Figura anexo 2.1-** Archivo de descarga del paquete de instalación de MinGW.