Universidad de Burgos

Escuela Politécnica Superior

Ingeniería Informática

Área de Lenguajes y Sistemas Informáticos

X-RayDetector: Detección automática de defectos en piezas metálicas mediante análisis de radiografías

Adrián González Duarte y Joaquín Bravo Panadero

Directores: Dr. César I. García Osorio
José Francisco Díez Pastor

Burgos, junio de 2013

Resumen

La inspección de defectos es una tarea muy importante para asegurar la seguridad y la fiabilidad de los procesos industriales. Una de las últimas tendencias en la inspección y control de calidad de los procesos industriales son las técnicas de prueba no destructiva (NDT - Non-Destructive Testing). En estas técnicas se evalúan los defectos superficiales o internos a través de distintos métodos como son la radiografía y la ecografía, entre otros.

La radiografía se ha mostrado como uno de los métodos más efectivos a la hora de identificar defectos en procesos de inyección de material en molde. Este método se basa en que las zonas con algún tipo de defecto absorben distinta cantidad de energía que el resto de zonas, lo que ocasiona regiones más oscuras o claras en la imagen.

La etapa de interpretación de los defectos en imágenes de radiografía es un proceso costoso en tiempo y subjetivo, produciéndose contradicciones entre las evaluaciones de los distintos inspectores de la imagen. Este trabajo es la segunda iteración de un proyecto que pretende desarrollar un método para la detección y clasificación de defectos en imágenes de radiografía de piezas de magnesio, que acelere el proceso de inspección de fallos y elimine la ambigüedad y subjetividad del proceso manual.

Descriptores

Detección de defectos, minería de datos, procesamiento digital de la imagen, visión artificial, aprendizaje automático, radiografías

Abstract

The defect inspection is an important task to ensure the safety and reliability of industrial processes. Some of the latest trends in inspection and quality control of industrial processes are the non-destructive testing techniques (NDT - Non-Destructive Testing). These techniques are evaluated on the surface or internal defects through different methods such as radiography and ultrasound, among others.

Radiography has proven to be one of the most effective techniques to identify defects in the process of injection-molded material. This method is based on the fact that areas with some kind of defect absorb different amounts of energy than other areas, resulting in darker or lighter regions in the image.

The stage of defects' interpretation in radiographic images is a subjective and time consuming process, resulting in contradictions between different operators examining the same. This work is the second iteration of a project for the detection and classification of defects in radiographic images of magnesium parts, which aim is to speed up the process of defect detection and the elimination of the ambiguity and subjectivity of the human inspection.

Keywords

Defect detection, Data Mining, digital image processing, computer vision, Machine Learning, radiography

Índice general

Índice de figuras

Índice de tablas

Agradecimientos

Queremos aprovechar este pequeño espacio para dar las gracias a todas las personas que han hecho posible el desarrollo de este proyecto.

En primer lugar, queremos dar las gracias a nuestros tutores, César García Osorio y José Francisco Díez Pastor por la ayuda mostrada y por haber confiado en nosotros desde el principio. El magnífico trato recibido por su parte tanto en el plano docente, como personal, han hecho posible la realización de este proyecto, guiándonos en todo momento y transmitiéndonos los conocimientos necesarios, además de haber colaborado activamente en el desarrollo de todo el proyecto. Ha sido una experiencia muy grata el haber podido trabajar con ellos.

En segundo lugar, también queremos dar las gracias a los profesores de la Universidad de Burgos, ya que sin su dedicación y saber hacer nos hubiese sido imposible haber adquirido los conocimientos necesarios para la realización de este proyecto y, por consiguiente, formarnos como ingenieros.

Tampoco queremos olvidarnos de nuestros compañeros que nos han animado y nos han ayudado con algún pequeño detalle del proyecto, aportando ideas y otros conocimientos.

Por último, queremos agradecer a nuestras familias y amigos el apoyo recibido durante todos estos años. Sin ellos, no nos hubiera sido posible llegar hasta aquí.

1. OBJETIVOS DEL PROYECTO

Los objetivos principales del proyecto son fundamentalmente dos:

1. El rediseño de la aplicación anterior, buscando un diseño mucho más modular y reutilizable, refactorizando el código que reutilicemos, buscando un mejor estilo para favorecer su compresión y reutilización, y que facilita la extensión futura con nueva funcionalidad. Este objetivo aunque no ha añadido funcionalidad al proyecto previo ha supuesto una parte importante del tiempo dedicado al proyecto, el resultado de este objetivo además de facilitar la reutilización de la funcionalidad de la versión previa ha sido un paso necesario para acometer su ampliación.
2. Ampliar la funcionalidad de la aplicación previa:
   a) Mejorando el rendimiento de la misma preparándola para aprovechar los procesadores actuales de más de un núcleo de proceso.
   b) Mejorando la precisión de las técnicas utilizadas para la localización de defectos.
   c) Añadiendo algoritmos de segmentación que permitan obtener una localización más precisa de los defectos de la imagen, así como aplicar cálculos de características geométricas.
   d) Implementando los algoritmos de selección de características más relevantes.
   e) Implementando medidas de precisión, como *precision & recall*.

En el proyecto se van a implementar los algoritmos de extracción de características más relevantes o significativos que hemos encontrado, prestando especial atención al diseño para que sea fácil la inclusión de nuevos algoritmos en el futuro. Esto posibilitará que el proyecto crezca en el tiempo y amplíe su capacidad.

1.1 Objetivos Técnicos

Este proyecto se va a desarrollar utilizando el paradigma de la *Orientacion a Objetos* el cual nos permitirá un diseño fácilmente comprensible por futuros desarrolladores que deseen proseguir con el presente trabajo.

El lenguaje de programación escogido para el proyecto es *Java 7*. El lenguaje utilizado venía dado por el lenguaje en que estaba desarrollada la aplicación previa. Pero de no haber sido así, posiblemente hubiera sido el lenguaje elegido debido al conocimiento del mismo, su sencillez, portabilidad, extensa documentación y la posibilidad de utilizar la librería de *Weka* e *ImageJ*. La completa *API (Application Programming Interface)* con la que cuenta y los conocimientos adquiridos durante la carrera sobre este lenguaje de programación harán posible que el desarrollo se base en el estudio e implementación de los algoritmos evitando retrasos por tener que aprender un nuevo lenguaje de programación

El lenguaje de modelado escogido ha sido *UML* (*Unified Modeling Language*) que se trata de un lenguaje unificado y muy extendido en el diseño de aplicaciones Orientadas a Objetos (OO).

Para la creación de los diagramas se utilizará *Astah*, se trata de una herramienta para el modelado de diagramas *UML*. Al estar enfocado a la OO posibilita todo tipo de diagramas de una forma cómoda y rápida. Todos los diagramas creados a lo largo del proyecto se realizarán con dicho programa.

Para resolver algunos de los problemas comunes a los que habrá que enfrentarse se utilizarán diversos patrones de diseño [5] estudiados que posibilitarán ofrecer una solución única estándar sobre problemas comunes que puedan surgir. La aplicación de patrones consigue diseños de calidad y, en consecuencia, mejores resultados en la fase de implementación.

También, trabajaremos con programación multihilo, buscando el poder aprovecharnos de las posibilidades que ofrecen los procesadores actuales, con más de un núcleo de proceso. Con esto, aumentaremos notablemente el rendimiento.

Para la memoria se va a utilizar LaTeX [26]. Las ventajas que ofrece respecto a otros sistemas de composición de textos (como los clásicos *WYSIWYG* [30]) son muchas, entre ellas nos permitirá la creación de una documentación uniforme, es decir, la salida que se obtenga será la misma con independencia del dispositivo o sistema operativo empleado para su visualización o impresión.

Dado que nunca se ha trabajado con LaTeX ha sido necesaria la utilización de documentación [27], manuales [28] y [29] que han facilitado el aprendizaje del mismo. Como plantilla para la memoria se utilizará la adaptación de la de la Universidad de Deusto llevada a cabo por Álvar Arnáiz para elaborar la documentación de su proyecto [32].

Para trabajar con las radiografías se ha elegido la librería ImageJ. Se trata de un programa de procesamiento de imagen digital desarrollado en Java que permite analizar y procesar imágenes, mediante la utilización de diversas técnicas como filtros e histogramas.

1.2 Objetivos personales

Además de los objetivos propios de la realización de la aplicación, también se pretende conseguir una serie de objetivos personales. Nos gustaría poder poner en práctica todos los conocimientos teóricos adquiridos durante estos años de carrera. Además, con la realización de este proyecto queremos adquirir nuevos conocimientos en áreas tan diversas como la gestión de proyectos y el uso de metodologías ágiles, la inteligencia artificial y la visión por computador, la minería de datos, el uso de sistemas de control de versiones, etc. Por último, queremos llevar a cabo con éxito la realización de este proyecto siendo capaces de planificarnos y trabajar en equipo.

2. CONCEPTOS TEÓRICOS

En este apartado se van a describir, por una parte, conceptos básicos de la visión por computador, comenzado con nociones básicas del tratamiento digital de imágenes como la Adquisición (2.2.1), el preprocesamiento (2.2.2) y la segmentación (2.2.3). A continuación, se describirán distintas técnicas de análisis de imágenes como la extracción de características (2.2.4) y el reconocimiento e interpretación (2.2.5). Por otra parte se hará una breve introducción a cómo se utilizan técnicas de visión artificial en los ensayos no destructivos y una descripción más pormenorizada de estás técnicas dentro de nuestro proyecto (2.3).

2.1 Descripción del problema: Ensayos no destructivos

Los ensayos no destructivos [52] son pruebas que, practicadas sobre un material, no alteran de forma permanente sus propiedades físicas, químicas, mecánicas o dimensionales. Los ensayos no destructivos implican un daño imperceptible o nulo.

Los diferentes métodos de ensayos no destructivos se basan en la aplicación de fenómenos físicos tales como ondas electromagnéticas, acústicas, elásticas, emisión de partículas subatómicas, capilaridad, absorción y cualquier tipo de prueba que no implique un daño considerable a la muestra examinada.

Los datos aportados por este tipo de ensayos suelen ser menos exactos que los de los ensayos destructivos. Sin embargo, suelen ser más baratos, ya que no implican la destrucción de la pieza a examinar. En ocasiones los ensayos no destructivos buscan únicamente verificar la homogeneidad y continuidad del material analizado, por lo que se complementan con los datos provenientes de los ensayos destructivos.

Uno de los aspectos más importantes de cualquier método de ensayo no destructivo es que todo el personal debe estar entrenado, cualificado y certificado. El personal debe estar familiarizado con las técnicas, el equipamiento, el objeto a ensayar y cómo interpretar los resultados.

El objetivo de este proyecto es proporcionar una herramienta que automatice en la medida de lo posible (o que en última instancia, sirva de asistencia al operador humano) el proceso de ensayo radiográfico.

2.1.1 Radiografía

Una radiografía [55] es una imagen registrada en una placa o película fotográfica, o de forma digital en una base de datos. La imagen se obtiene al exponer al receptor de imagen radiográfica a una fuente de radiación de alta energía, comúnmente rayos X o radiación gamma procedente de isótopos radiactivos. Al interponer un objeto entre la fuente de radiación y el receptor, las partes más densas aparecen con diferentes tonos dentro de una escala de grises, en función inversa a la densidad del objeto. Por ejemplo, si la radiación incide directamente sobre el receptor, se registra un tono negro.

Sus usos pueden ser tanto médicos, para detectar fisuras en huesos, como industriales en la detección de defectos en materiales y soldaduras, tales como grietas, poros, rebabas, etc.

La radiografía se usa para ensayar una variedad de productos, tales como objetos de fundición, objetos forjados y soldaduras. Es también muy usada en la industria aeroespacial para la detección de grietas (fisuras) en las estructuras de los aviones, la detección de agua en las estructuras tipo panal y detección de objetos extraños. Los objetos a ensayar se exponen a rayos X o gamma y se procesa un film o se visualiza digitalmente. El personal de ensayos radiográficos instala, expone y procesa la película o digitalmente procesa las señales e interpreta las imágenes de acuerdo con códigos.

2.1.2 Ventajas del ensayo radiográfico

Las ventajas del ensayo radiográfico [55] son, entre otras, las siguientes:

1. Puede usarse con la mayoría de los materiales.
2. Puede usarse para proporcionar un registro visual permanente del objeto ensayado o un registro digital con la subsiguiente visualización en un monitor de computadora.
3. Puede revelar algunas discontinuidades dentro del material.
4. Revela errores de fabricación y a menudo indica la necesidad de acciones correctivas.

2.1.3 Limitaciones del ensayo radiográfico

Las limitaciones de la radiografía [55] incluyen consideraciones físicas y económicas.

1. Deben seguirse siempre los procedimientos de seguridad para las radiaciones.
2. La accesibilidad puede estar limitada. El operador debe tener acceso a ambos lados del objeto a ensayar.
3. Las discontinuidades que no son paralelas con el haz de radiación son difíciles de localizar.
4. La radiografía es un método caro de ensayo.
5. Es un método de ensayo que consume mucho tiempo. Después de tomar la radiografía, la película debe ser procesada, secada e interpretada (aunque este problema desaparece cuando la imagen de rayos X se registra digitalmente).
6. Algunas discontinuidades superficiales pueden ser difíciles, si no imposible, de detectar.

2.1.4 Objetivos del ensayo radiográfico

El objetivo del ensayo radiográfico [55] es asegurar la confiabilidad del producto. Esto puede lograrse sobre la base de los siguientes factores.

1. La radiografía permite al técnico ver la calidad interna del objeto ensayado o evidencia la configuración interna de los componentes.
2. Revela la naturaleza del objeto ensayado sin perjudicar la utilidad del material.

3. Revela discontinuidades estructurales, fallas mecánicas y errores de montaje.

La realización del ensayo radiográfico es sólo una parte del procedimiento. Los resultados del ensayo deben ser interpretados de acuerdo con normas de aceptación, y luego el objeto ensayado es aceptado o rechazado.

2.1.5 Principios del ensayo radiográfico

Los rayos X y gamma [55] tienen la capacidad de penetrar los materiales incluso los materiales que no transmiten la luz. Al pasar a través de un material, algunos de esos rayos son absorbidos. La cantidad de radiación que se transmite a través de un objeto ensayado varía dependiendo del espesor y densidad del material y del tamaño de la fuente que se use. La radiación transmitida a través del objeto produce una imagen radiográfica. El objeto ensayado absorbe radiación, pero hay menos absorción donde el objeto es más fino o donde se presenta un vacío. Las porciones más gruesas del objeto o las inclusiones más densas se verán como imágenes más claras en la radiografía porque aumenta el espesor y la absorción es mayor.

2.2 Visión por computador

2.2.1 Adquisición de la imagen

La primera etapa del proceso es la adquisición de la imagen. Para ello se necesitarán dos elementos:

- Un sensor de imágenes, es decir, un dispositivo físico sensible a una determinada banda del espectro de energía electromagnética (como las bandas de rayos X, ultravioleta, visible o infrarrojo). En nuestro caso será un sistema de rayos X.
- Un digitalizador, dispositivo que permitirá convertir la señal de salida del sensor a forma digital.

En este proyecto no se está realizando la adquisición de la imagen, ya que trabajamos con imágenes cedidas por el Grupo Antolín.

2.2.2 Preprocesamiento

El preprocesamiento de la imagen es una etapa que consiste en reducir la información de la misma, de forma que pueda ser interpretada por una computadora, facilitando así la posterior fase de análisis.

Se utiliza un conjunto de técnicas que, aplicadas a las imágenes digitales, mejoran su calidad o facilitan la búsqueda de información. A partir de una imagen origen, se obtiene otra imagen final cuyo resultado sea más adecuado para una aplicación específica, optimizando ciertas características de la misma que hagan posible realizar operaciones de procesado sobre ella.

A continuación se explican algunas de las técnicas que comprenden esta etapa.

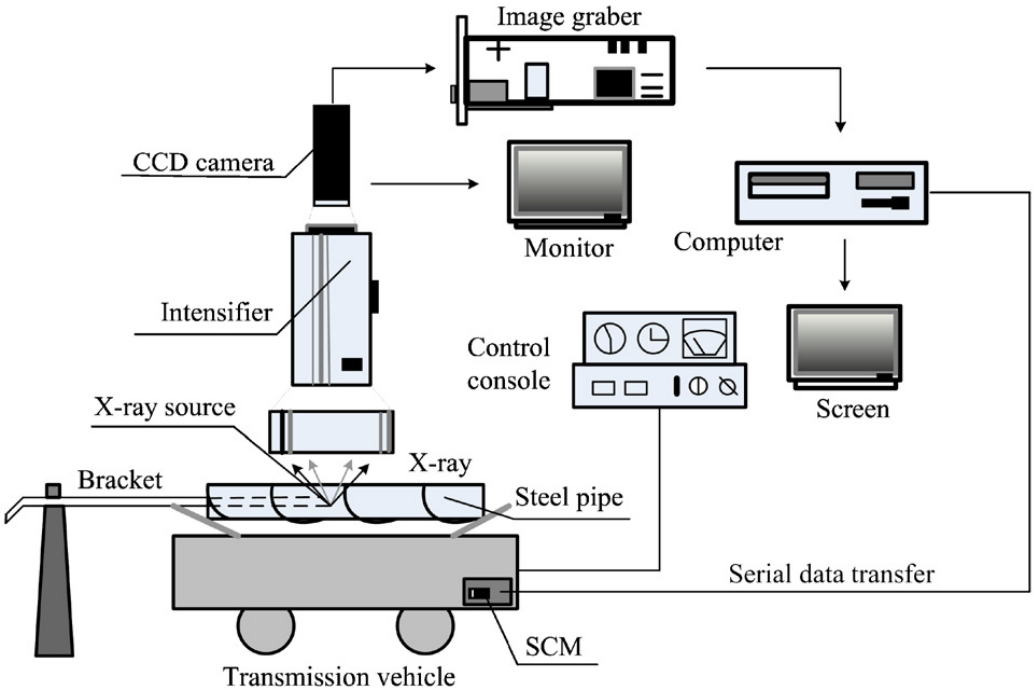

Figura 2.1.: Proceso de adquisición de imágenes de radiografía [52]

■ Binarización

La binarización de una imagen consiste en un proceso mediante el cual los valores de gris de una imagen quedan reducidos a dos (que podrían interpretarse como falso y verdadero). En una imagen digital, estos valores pueden representarse por los valores 0 y 1 o, más frecuentemente, por los colores negro (valor de gris 0) y blanco (valor de gris 255).

Para hacer esto, primero se debe convertir la imagen a escala de grises. Después hay que fijar un valor umbral entre 0 y 255. Una vez que se tenga dicho umbral, se convertirán a 255 todos los valores de la imagen superiores al umbral, mientras que los inferiores se convertirán a 0. El resultado será una imagen en blanco y negro que permitirá realizar tareas como la detección de contornos, separar regiones u objetos de interés del resto de la imagen, etc.

La binarización es, además, un método de segmentación. Hablaremos más sobre este tema en la sección 2.2.3.

■ Saliency

El «Saliency Map» o «Mapa de Prominencia» [38] es un mapa topográfico que permite representar la prominencia visual de una determinada imagen.

Uno de los mayores problemas de la percepción es la sobrecarga de información. Se hace necesario identificar qué partes de la información disponible merecen ser seleccionadas para ser analizadas y qué partes deben descartarse. Este algoritmo busca solucionar este problema.

Figura 2.2.: Ejemplo de Saliency Map. La imagen original a la izquierda, con el correspondiente saliency map a la derecha

Koch y Ulman propusieron en 1985 [24] que las diferentes características visuales que contribuyen a la selección de atención ante un estímulo (color, orientación, movimiento, etc.) fueran combinadas en un único mapa topográfico, el Saliency Map, que integraría la información normalizada de los mapas de características individuales en una medida global de visibilidad.

La saliencia de una posición dada es determinada principalmente por cómo de diferente es dicha localización de las que la rodean, en color, orientación, movimiento, profundidad, etc.

La implementación del mapa de saliencia usada en este proyecto está basada en la variante descrita en el artículo Human Detection Using a Mobile Platform and Novel Features Derived From a Visual Saliency Mechanism [37].

2.2.3 Segmentación

Además de calcular ciertas características de las imágenes, como hemos visto en el apartado anterior, se hace necesaria realizar una segmentación de la imagen, para aumentar la precisión de la detección de los defectos.

La segmentación de una imagen [53] consiste en particionar esa imagen en múltiples segmentos (conjuntos de píxeles). El objetivo es simplificar y/o cambiar la representación de una imagen en algo que sea más significativo o fácil de analizar. Se suele usar típicamente para localizar objetos y bordes. Más precisamente, la segmentación de una imagen es el proceso de asignar una etiqueta a cada píxel de una imagen, con lo que los píxeles que tengan la misma etiqueta compartirán ciertas características.

De entre todos los posibles métodos de segmentación que existen, nosotros hemos usado el denominado *Thresholding* [56]. Es uno de los métodos más simples. Está basado en considerar un valor, llamado **umbral**, que se usa para convertir una imagen en escala de grises a una binaria (es

decir, binarizar la imagen). La clave está, por tanto, en el valor umbral. De acuerdo a la taxonomía definida por Sezgin and Sankur [43], existen varias opciones:

- Basados en la forma del histograma.
- Basados en clustering.
- Basados en entropía.
- Basados en atributos de objetos.
- Métodos espaciales.
- Métodos locales.

De todos estos, nosotros hemos usado los locales. Estos métodos se basan en adaptar el valor umbral en cada píxel, de acuerdo a las características del vecindario de ese píxel. El radio del vecindario es un parámetro que afecta al funcionamiento del método.

Hay varios de estos métodos. A continuación, vamos a describir los que hemos probado en este proyecto.

■ MidGrey

Este método selecciona el umbral de acuerdo a la media del máximo y mínimo valor de la distribución local de escala de grises.

Por lo tanto, el umbral viene dado por la siguiente fórmula [3]:

$$T = \left(\frac{\text{máx} + \text{mín}}{2}\right) - c$$

Donde c es una constante que sirve para afinar el método. Por defecto, es cero.

Para determinar a qué región pertenece el píxel, se comprueba con el umbral. Si es mayor que éste último, el píxel pertenece al objeto. Si no, pertenece al fondo.

■ Mean

En este caso, el umbral es la media de la distribución local en escala de grises [3]. Por lo tanto, para determinar a qué región pertenece el píxel, se compara con la media de la región de vecinos (menos el parámetro c, en caso de que se especifique). Si es mayor, es parte del objeto. Si no, es parte del fondo.

2.2.4 Descriptores de regiones

Una vez realizada la etapa de preprocesamiento, la imagen ya estará lista para ser analizada. Nosotros vamos a trabajar directamente con los píxeles de la imagen, a diferencia de otros métodos de análisis, que extraen otros tipos de atributos. Para analizar las características de la imagen, se utilizarán los descriptores que se explican a continuación:

- Descriptores simples.
- Descriptores de textura.
- Descriptores geométricos.

■ Descriptores simples

También se les conoce como características estándar o de primer orden [41]. Son medidas que se calculan a partir de los valores de gris originales de la imagen y su frecuencia, como la media, varianza, desviación estándar, etc. En estas medidas no se considera la relación de co-ocurrencia entre los píxeles.

Las características más comunes y que se han usado en este proyecto son:

Media

Se calcula el promedio de los niveles de intensidad de todos los píxeles de la imagen. Esta es una medida útil ya que nos permite determinar de forma sencilla la claridad de la imagen. Si la media es alta, la imagen será más clara, mientras que si la media es baja, será más oscura.

Desviación estándar

Es una medida de dispersión que nos indica cuánto se alejan los valores respecto a la media. Nos sirve para apreciar el grado de variabilidad entre los valores de intensidad de los píxeles de una región.

Primera y segunda derivadas

Se utilizan operadores de detección de bordes [34] basados en aproximaciones de la primera y segunda derivada de los niveles de grises de la imagen. La primera derivada del perfil de gris será positiva en el borde de entrada de la transición entre una zona clara y otra oscura. En el borde de salida será negativa, mientras que en las zonas de nivel de gris constante será cero. El módulo de la primera derivada podrá utilizarse, por lo tanto, para detectar la presencia de un borde en una imagen. En cuanto a la segunda derivada, será positiva en la parte de la transición asociada con el lado oscuro del borde, negativa en la parte de la transición asociada con el lado claro y cero en las zonas de nivel de gris constante. El signo de la segunda derivada nos permitirá determinar si un píxel perteneciente a un borde está situado en el lado oscuro o claro del mismo.

■ Descriptores de textura

La texturas son propiedades asociadas a las superficies, como rugosidad, suavizado, granularidad, regularidad. En el campo de las imágenes, significa la repetición espacial de ciertos patrones sobre una superficie.

Otra definición de la textura podría ser la variación entre píxeles en una pequeña vecindad de una imagen. Alternativamente, la textura puede describirse también como un atributo que representa la distribución espacial de los niveles de intensidad en una región dada de una imagen digital.

El análisis de la textura de las imágenes nos ofrecerá datos útiles para nuestro trabajo. Hemos utilizado los siguientes descriptores:

Características de Haralick

Siguiendo la propuesta de Haralick [20], se extrae información de textura de la distribución de los valores de intensidad de los píxeles. Dichos valores se calculan utilizando matrices de coocurrencia que representan información de textura de segundo orden.

Haralick propuso un conjunto de 14 medidas de textura basada en la dependencia espacial de los tonos de grises. Esas dependencias están especificadas en la matriz de co-ocurrencia espacial (o de niveles de gris). Se sigue la descripción de [41] para calcular dicha matriz.

La matriz de co-ocurrencia, una vez normalizada, tiene las siguientes propiedades:

- Es cuadrada.
- Tiene el mismo número de filas y columnas que el número de bits de la imagen. Con una imagen de 8 bits ($2^8 = 256$ posibles valores) la matriz de co-ocurrencia será de $256x256$, es decir, 65536 celdas.
- Es simétrica con respecto a la diagonal.
- Los elementos de la diagonal representan pares de píxeles que no tienen diferencias en su nivel de gris. Si estos elementos tienen probabilidades grandes, entonces la imagen no muestra mucho contraste, ya que la mayoría de los píxeles son idénticos a sus vecinos.
- Sumando los valores de la diagonal tenemos la probabilidad de que un píxel tenga el mismo nivel de gris que su vecino.
- Las líneas paralelas a la diagonal separadas por una celda, representan los pares de píxeles con una diferencia de un nivel de gris. De la misma manera sumando los elementos separados dos celdas de la diagonal, tenemos los pares de píxeles con dos valores de grises de diferencia. A medida que nos alejamos de la diagonal la diferencia entre niveles de grises es mayor.
- Sumando los valores de estas diagonales secundarias (y paralelas a la diagonal principal) obtenemos la probabilidad de que un píxel tenga 1, 2, 3, etc niveles de grises de diferencia con su vecino.

Una vez construida la matriz de co-ocurrencia, de ella pueden derivarse diferentes medidas. Se obtendrán matrices para las direcciones 0º, 90º, 180º y 270º y para distancias 1, 2, 3, 4 y 5. Para cada una de estas distancias se calculará un vector con las medias de las cuatro direcciones y otro con los rangos. Las características a calcular a partir de la matriz son las siguientes:

1. **Segundo Momento Angular**

   Mide la homogeneidad local. Cuanto más suave es la textura, mayor valor toma. Si la matriz de co-ocurrencia tiene pocas entradas de gran magnitud, toma valores altos. Es baja cuando todas las entradas son similares [6].

$$f_1 = \sum_{i=1}^{N_g} \sum_{j=1}^{N_g} p(i,j)^2$$

   $p(i,j)$ es el valor de la matriz de coocurrencia en la fila i y la columna j N_g es la dimensión de la matriz

2. **Contraste**

   Es lo opuesto a la homogeneidad, es decir, es una medida de la variación local en una imagen. Tiene un valor alto cuando la región tiene un alto contraste.

$$f_2 = \sum_{n=0}^{N_g-1} n^2 \sum_{i=1}^{N_g} \sum_{j=1}^{N_g} p(i,j)$$

3. **Correlación**

   Mide las dependencias lineales de los niveles de grises, la similitud entre píxeles vecinos. Un objeto tiene mayor correlación dentro de él que con los objetos adyacentes. Píxeles cercanos están más correlacionados entre sí que los píxeles más distantes.

$$f_3 = \frac{\sum_i \sum_j (i,j) \cdot p(i,j) - v_x v_y}{\sigma_x \sigma_y}$$

   Donde $v_x, v_y, \sigma_x, \sigma_y$ son las medias y desviaciones estándar de p_x y p_y, las funciones de densidad de probabilidad parcial.

4. **Suma de cuadrados**

   Es la medida del contraste del nivel de gris.

$$f_4 = \sum_{i=1}^{N_g} \sum_{j=1}^{N_g} (i-j)^2 \cdot p(i,j)$$

5. **Momento Diferencial Inverso**

   También llamado homogeneidad, es más alto cuando la matriz de co-ocurrencia se concentra a lo largo de la diagonal. Esto ocurre cuando la imagen es localmente homogénea de acuerdo al tamaño de la ventana.

$$f_5 = \sum_i \sum_j \frac{1}{1+(i-j)^2} \cdot p(i,j)$$

6. **Suma promedio**

$$f_6 = \sum_{i=2}^{2N_g} i \cdot p_{x+y}(i)$$

7. **Suma de Entropías**

$$f_7 = \sum_{i=2}^{2N_g} (i - f_8)^2 \cdot p_{x+y}(i)$$

8. **Suma de Varianzas**

$$f_8 = -\sum_{i=2}^{2N_g} p_{x+y}(i) \log(p_{x+y}(i))$$

9. **Entropía**

   Es alta cuando los elementos de la matriz de co-ocurrencia tienen valores relativamente iguales. Es baja cuando los elementos son cercanos a 0 ó 1.

$$f_9 = -\sum_{i=1}^{N_g} \sum_{j=1}^{N_g} p(i,j) \log(p(i,j))$$

10. **Diferencia de Varianzas**

$$f_{10} = \sum_{i=0}^{N_g-1} i^2 p_{x-y}(i)$$

11. **Diferencia de Entropías**

$$f_{11} = -\sum_{i=0}^{N_g-1} p_{x-y}(i) \log(p_{x-y}(i))$$

12. **Medidas de Información de Correlación 1**

$$f_{12} = \frac{HXY - HXY1}{\text{máx}(HX, HY)}$$

   Donde:

$$HXY = -\sum_{i=1}^{N_g} \sum_{j=1}^{N_g} p(i,j) \log\big(p(i,j)\big)$$

$$HXY1 = -\sum_{i=1}^{N_g} \sum_{j=1}^{N_g} p(i,j) \log\big(p_x(i) p_y(j)\big)$$

$$HXY2 = -\sum_{i=1}^{N_g} \sum_{j=1}^{N_g} p_x(i) p_y(j) \log\big(p_x(i) p_y(j)\big)$$

13. **Medidas de Información de Correlación 2**

$$f_{13} = \big(1 - \exp(-2|HXY2 - HXY|)\big)^{1/2}$$

14. **Coeficiente de Correlación Máxima**

$$f_{14} = \sqrt{\lambda_2}$$

Donde λ_2 es el segundo valor propio de la matriz Q definida como:

$$Q(i,j) = \sum_k \frac{p(i,k)p(j,k)}{p_x(i)p_y(i)}$$

Local Binary Patterns

Los Local Binary Patterns (LBP) [51] son un tipo de característica usado para la clasificación de texturas. Fueron descritos por primera vez en 1994 [44].

Debido a su poder de discriminación y su simplicidad de cálculo, se ha convertido en un método popular que se usa en varios tipos de aplicaciones [17].

Este operador de textura etiqueta los píxeles de una imagen comparando los valores de intensidad de los píxeles de una vecindad de 3x3 con el del píxel central.

Cuando el valor del píxel vecino es mayor que el del píxel central, se escribe «1». En caso contrario, se escribe «0». El resultado es un número binario de 8 dígitos, que suele convertirse a decimal por comodidad o para mayor facilidad de cálculo. Este número recibe también el nombre de «patrón».

Luego se realiza un histograma que contendrá la frecuencia con la que se ha producido cada patrón. Dicho histograma podrá utilizarse como descriptor de textura.

Posteriormente se ha extendido el uso de diferentes tamaños, no sólo a ocho puntos, sino a muestreos circulares donde la bilinealidad se consigue con la interpolación de los valores de los píxeles, lo que permite utilizar cualquier radio y por lo tanto cualquier número de píxeles vecinos.

Para reducir la longitud del vector de características se utilizan los patrones uniformes. Un local binary pattern es uniforme si el patrón contiene un máximo de dos transiciones a nivel de bit, de «0» a «1» o viceversa.

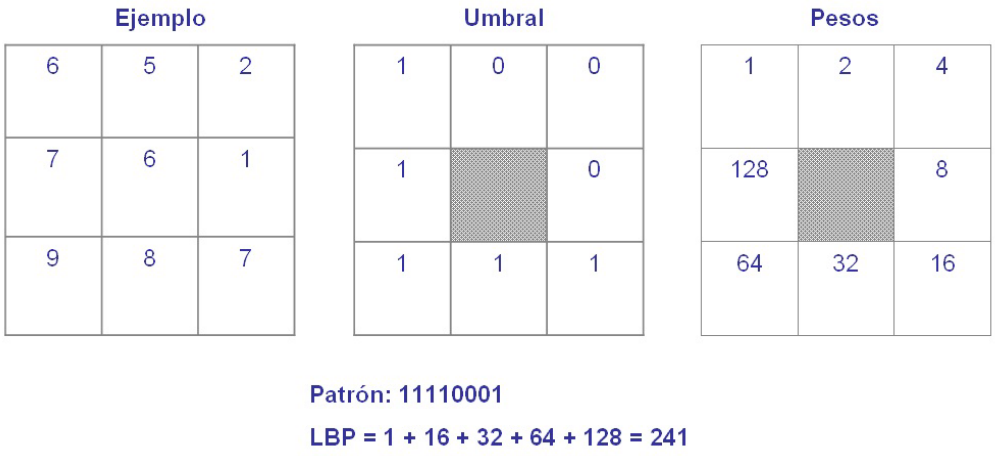

Figura 2.3.: Ejemplo del funcionamiento de los Local Binary Patterns [17]

Por ejemplo, los patrones 00000000 (0 transiciones), 01110000 (2 transiciones) y 11001111 (2 transiciones) son uniformes mientras que los patrones 11001001 (4 transiciones) y 01010010 (6 transiciones) no lo son. El número de transiciones se guarda en un valor llamado medida de uniformidad U.

En el cálculo de los LBP, se utiliza una etiqueta para cada uno de los patrones uniformes, mientras que todos los patrones no uniformes son agrupados en una sola etiqueta. Por ejemplo, cuando se usa una vecindad (8,R) (donde R es el radio), hay un total de 256 patrones, 58 de los cuales son uniformes, lo cual produce un total de 59 etiquetas diferentes. Todo esto dará como resultado un histograma con 59 intervalos.

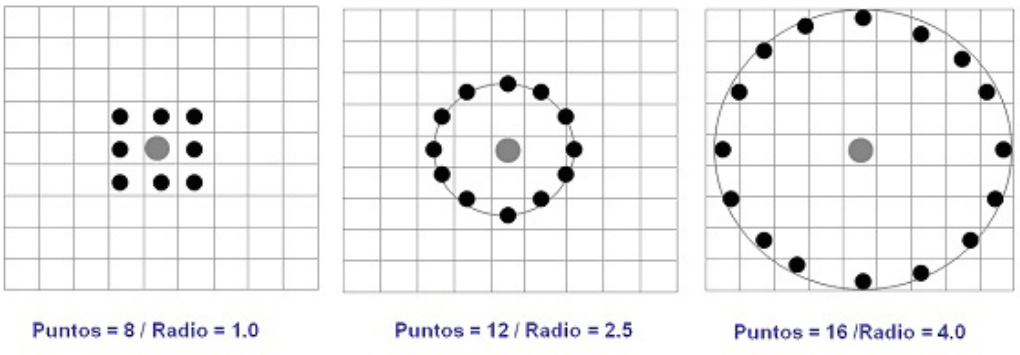

Figura 2.4.: Vecindades de distinto tamaño en los LBP [17]

■ Características geométricas

Además de los descriptores de regiones que ya hemos visto, pensamos en la posibilidad de realizar algunos cálculos de características geométricas sobre las regiones segmentadas, con la mirada puesta en poder clasificar mediante ellas a los defectos en distintos tipos.

Las características geométricas que hemos calculado son [1]:

- **Área:** es la superficie de la región de interés medida en píxeles cuadrados.
- **Perímetro:** es la longitud del límite exterior de la región.
- **Circularidad:** descriptor de forma dado por la fórmula $\frac{4\pi \times área}{perímetro^2}$. Un valor de 1 indica que se trata de un círculo perfecto. Según se acerca a cero, indica que la forma es cada vez más alargada.
- **Redondez:** descriptor de forma dado por la fórmula $\frac{4\pi \times área}{\pi \times semieje\_mayor^2}$. Es el inverso del cociente entre el semieje mayor y el menor de la mejor elipse que puede ser dibujada en la región.
- **Semieje mayor:** semieje mayor de la mayor elipse que puede ser dibujada en la región.
- **Semieje menor:** semieje menor de la mayor elipse que puede ser dibujada en la región.
- **Ángulo:** es el ángulo entre el semieje mayor y una línea paralela al eje x.
- **Distancia Feret:** también llamada diámetro de Feret. Es la máxima distancia entre dos puntos cualesquiera de la región.

2.2.5 Reconocimiento e interpretación de imágenes

Una vez obtenidas las características de la imagen, el siguiente paso será reconocer dichos datos e interpretarlos. Para ello será necesario entrenar un clasificador. Una vez entrenado, podrá predecir dónde estarán los defectos que buscamos.

Un clasificador [50] es un elemento que, tomando un conjunto de características como entrada, proporciona a la salida una etiqueta de clase. En nuestro caso, el atributo de clase sería el «defecto», que podría tomar dos valores: «verdadero» o «falso». En el caso de que sea una regresión, en vez de una clasificación de clases nominales, los valores que devolverá el clasificador serán 0 y 1.

Utilizaremos el clasificador por su capacidad de aprender a partir de imágenes de ejemplo y de generalizar este conocimiento para que se pueda aplicar a nuevas y diferentes imágenes.

Para construir el clasificador utilizaremos un conjunto de imágenes etiquetadas. Para etiquetarlas, se creará una máscara de cada imagen en la que se marcarán a mano los defectos. Estas máscaras permitirán al clasificador saber qué partes de la imagen son defectos y cuales no. Es lo que se denomina ground truth en la literatura de procesamiento digital de imágenes.

Los clasificadores que hemos utilizado no pueden trabajar directamente con imágenes, sino con vectores de características, que serán las que se calculen a partir de las imágenes de ejemplo. Estos vectores de características se guardarán en ficheros *ARFF*, que tendrán una serie de atributos definidos en la cabecera, cada uno de ellos correspondiente a una característica. El fichero contendrá un conjunto de instancias, que son cada serie de valores que toman los atributos. Los ficheros *ARFF* son el formato propio de *Weka*, y en su estructura se pueden diferenciar las siguientes secciones:

- **@relation:** Los ficheros *ARFF* deben empezar con esta declaración en su primera línea (no puede haber líneas en blanco antes). Será una cadena de caracteres.
- **@attribute:** En esta sección hay que poner una línea por cada atributo que vaya a tener el conjunto de datos. Para cada atributo habrá que indicar su nombre y el tipo de dato. El tipo puede ser numeric, string, etc.
- **@data:** En esta sección se incluyen los datos. Cada columna se separa por comas y todas las filas deberán tener el mismo número de columnas, que coincidirá con el número de atributos declarados.

Al entrenar al clasificador obtendremos un modelo, el cual usaremos cuando queramos detectar los defectos de una nueva imagen.

Un clasificador puede ser, según los tipos de aprendizaje:

Supervisado: cuando se utilizan ejemplos previamente etiquetados.

No supervisado: cuando se utilizan patrones de entradas para los no se especifican los valores de sus salidas.

Por lo dicho anteriormente, hemos utilizado clasificadores supervisados. Nos hemos basado en el estudio con varios clasificadores que hicieron los alumnos que desarrollaron la versión de la aplicación que este proyecto esta mejorando y ampliando, así que hemos usado el que ellos consideraron mejor: *Bagging*.

Como algoritmo base hemos usado *REPTree*, que, por sus características, también lo hemos podido usar para la regresión.

■ REPTree

El algoritmo de *REPTree* [21] permite construir tanto un árbol de clasificación como un árbol de regresión, usando las medidas de *ganancia de información* y de *reducción de la varianza*. Puede podar los árboles generados usando la *poda de error reducido*. Además, está optimizado para la rápida ejecución. Sólo ordena valores para atributos numéricos una única vez.

La ganancia de información es una reducción de la entropía esperada (es decir, de la impureza de un conjunto de ejemplos), causada por la partición de los ejemplos de acuerdo al atributo [22]. La reducción de la varianza es una técnica que se usa para aumentar la precisión, disminuyendo el error que se produce cuando se entrena con diferentes conjuntos de datos.

Considera los valores desconocidos partiendo instancias en pedazos, como hace *C4.5*. Se puede especificar el mínimo número de instancias por hoja, profundidad máxima del árbol (útil cuando se utiliza la técnica de *boosting*), la mínima proporción de la varianza de los datos del conjunto de entrenamiento que se consideran para partir (sólo para clases numéricas), y el número de pliegues por poda [21].

■ Bagging

Bagging (de *Bootstrap Aggregating*)[22] es un algoritmo que busca combinar predicciones de un cierto tipo de modelo (en nuestro caso, *REPTree*) mediante votación o media, teniendo los modelos el mismo peso.

Lo que se hace es, teniendo un conjunto de datos, generar a partir de él un cierto número de conjuntos de datos mediante muestreo con reemplazamiento, para entrenar con ellos un cierto número de modelos. Después, se combinan las predicciones de todos los clasificadores entrenados para obtener la predicción final.

La combinación de estas predicciones se puede hacer con una media de las salidas (regresión) o con una votación de mayoría (clasificación).

Con esto conseguimos que métodos inestables, como árboles de decisión, mejoren su rendimiento.

2.3 Aplicación de la visión artificial a la detección y segmentación de defectos

2.3.1 Fundamentos teóricos de la versión inicial

En este apartado se describe la primera versión que realizamos de la detección de defectos. Esta primera versión es una versión mejorada de la versión final de los alumnos del año pasado.

Se ha continuado la misma idea que utilizaron los desarrolladores de la versión inicial, basada en el artículo «Automated Detection of Welding Defects without Segmentation» de Domingo Mery [35]. Los cambios realizadas en esta versión inicial con respecto a la versión de la que se partía han sido:

- Se ha realizado un nuevo diseño artquitectónico.
- Se ha cambiado parte del código de la interfaz.

- Se ha cambiado el proceso de entrenamiento.
- Ahora se utiliza un enfoque multihilo.

Preproceso

Para la realización del análisis de la imagen y su posterior detección de defectos, hay que partir una imagen original en escala de grises. A dicha imagen se la pasa un filtro *«Saliency Map»* (ver 2.2). Con todo esto, se obtiene la imagen original y la imagen Saliency que son analizadas paralelamente.

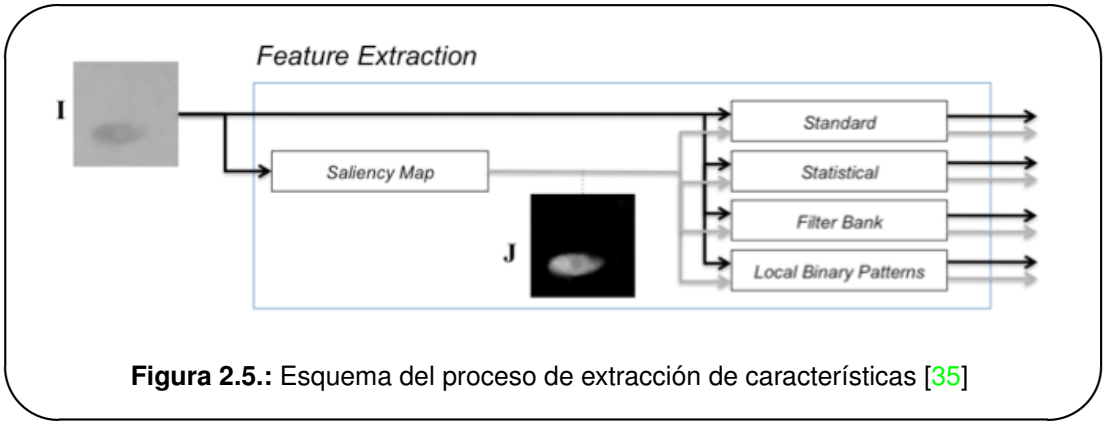

Figura 2.5.: Esquema del proceso de extracción de características [35]

Extracción de características y etiquetado de instancias

Antes de detectar cualquier tipo de defecto, es necesario entrenar un clasificador para que pueda predecir donde están los defectos buscados. Para ello se analizan un conjunto de imágenes de las que se crean máscaras en las que se pintan los defectos. Estas máscaras sirven para que el clasificador sepa qué partes de la imagen son defectos y cuáles no. En la imagen (ver figura 2.6) vemos un ejemplo de cómo son estas máscaras (también conocidas como *Ground Touch*).

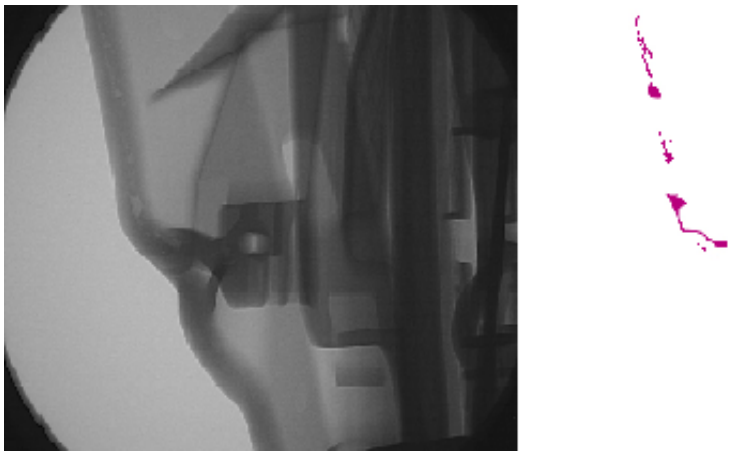

Figura 2.6.: Ejemplo de máscara junto a imagen original

Para analizar cada imagen se utiliza una ventana que recorre toda la región de interés analizando sus características. En el artículo original de Mery [35] sólo aparecía el número total de características, lo cual dificultó la identificación del número que había que calcular para cada tipo, por eso nosotros las hemos desglosado según su clase:

- **Características estándar:** 4 características.
- **Características de Haralick:** Se calculan 14 características, obteniendo 5 vectores de medias y 5 de rangos. $14 \times 10 = 140$ características.
- **Características LBP:** Se obtiene un histograma con 59 intervalos, es decir, 59 características.

El número final de características habrá que multiplicarlo por dos, ya que todas las características se calculan tanto para la imagen original como para la imagen con saliency map aplicado. Por lo tanto, el total sería:

(4 estándar + 140 haralick + 59 lbp)$\times$2 = 406 CARACTERÍSTICAS

Estas características serán las que se guarden en los ficheros *ARFF* que se utilicen para entrenar al clasificador, junto con la clase, que tendrá valor «true» si la instancia tiene defecto, o «false» si no. En caso de que se use regresión lineal serán 0 y 1. El defecto se identificará gracias a las máscaras mencionadas anteriormente.

La ventana irá analizando cada vez una región de la imagen, recorriendo todos y cada uno de sus píxeles, de los que extraerá las características que posteriormente se analizarán. Cada vez que la ventana se mueva y analice una nueva región, se creará una nueva instancia, que se corresponderá con una línea del fichero *ARFF* en el que se guardan las características. Nosotros hemos hecho que el tamaño de la ventana sea configurable, aunque por defecto se utilizará una ventana de 24$\times$24 píxeles, como en el artículo de Mery [35]. Utilizamos dos tipos de estrategias de desplazamiento:

- Ventana deslizante.
- Ventana aleatoria.

Ventana deslizante

La ventana comienza desde la esquina superior izquierda de la imagen y se va moviendo cada cierto porcentaje del tamaño de la ventana. Cuando llega al extremo derecho baja ese mismo porcentaje y comienza de nuevo desde el lado izquierdo. A diferencia del artículo, donde se usa siempre un salto de 4 píxeles, nosotros hemos hecho que se pueda seleccionar el tamaño del salto.

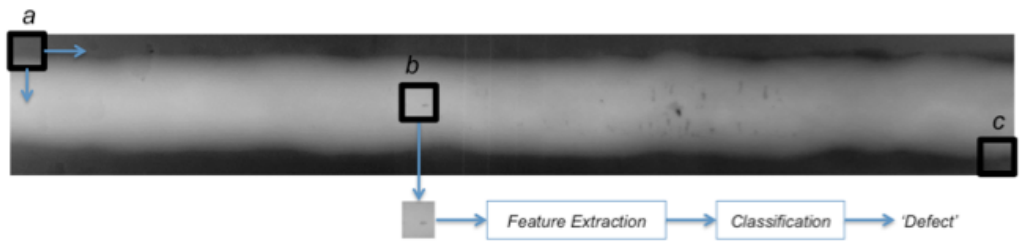

Figura 2.7.: Esquema de la ventana deslizante [35]

Ventana aleatoria

Se obtienen 300 muestras de ventanas por imagen, seleccionándolas aleatoriamente entre aquellas que tienen defecto y las que no.

■ Estrategias de etiquetado

Como hemos visto, se hace necesario determinar cuándo una ventana tiene defecto o no, ya que se necesita etiquetar las muestras de cada ventana como «defecto» o «no defecto». Los alumnos que desarrollaron la versión previa utilizaron una aproximación muy simple: consideraban que una ventana era defectuosa cuando, al poner esa ventana sobre la máscara coloreada manualmente, al menos un píxel de la misma estaba coloreado. Comprobamos que esto provoca falsos positivos, con lo que se pierde exactitud. Por lo tanto, decidimos implementar otras posibilidades:

Píxel central

En este caso, se considera una ventana defectuosa cuando el píxel central de la misma es defectuoso. Es un poco más preciso que la versión del año pasado, pero sigue sin ser demasiado buena.

Porcentaje de la ventana

En esta aproximación, se considera defectuosa una ventana cuando al menos un cierto porcentaje de la ventana contiene píxeles coloreados, es decir, defectuosos. Hemos obtenido resultados muy buenos con porcentajes que van desde el 50 % al 75 %.

Píxel central más región de vecinos

En este caso, no sólo consideramos el píxel central, si no que creamos una región cuadrada de 3×3 píxeles a su alrededor. Consideramos entonces una ventana como defectuosa cuando un cierto porcentaje de esta región de vecinos está coloreada. Obtenemos unos resultados muy parecidos a los de la anterior aproximación con porcentajes parecidos.

■ Detección de defectos

El proceso es prácticamente igual al entrenamiento del clasificador. Se genera una imagen Saliency Map a partir de la imagen original, al igual que antes, y a partir de las dos imágenes (la original y la Saliency) se analizan mediante una ventana deslizante.

Cada vez que la ventana se desplaza se genera una instancia que contiene todas las características analizadas. Esta instancia es utilizada por el modelo que predice si la ventana está situada sobre un defecto.

Si la ventana contiene un defecto, es marcada con un cuadrado verde y a cada uno de sus píxeles se le suma una unidad (este valor de los píxeles desde ahora será llamado factor).

Estos píxeles son almacenados en una matriz global del mismo tamaño que la imagen a analizar, por lo que según se desplaza la ventana deslizante se van actualizando todos los valores de los píxeles que contienen defectos.

Una vez finalizado el proceso de detección, se binariza la matriz resultante según su factor. Para este proceso en el artículo original [35] se utiliza un umbral de 24.

Este factor es descrito como el número de veces que ha sido marcado un píxel como defecto. Si un píxel tiene un factor menor que 24, dicho pixel se considera como no defecto. En cambio, si un píxel tiene un factor de 24 o más será considerado como defecto.

Dicho esto, en el proyecto utilizamos un factor variable. Por defecto el factor es de 8, pero tras la ejecución se puede variar para observar los cambios en la imagen en tiempo real.

Como ya se ha descrito, el proceso de binarización recae en el factor de un píxel. Si el factor es 8 o mayor, al pixel se le asigna un 1 (defecto), si es menor que 8 se le asigna un 0 (no defecto). Al tener una matriz de ceros y unos es sencillo obtener una imagen binaria con la región de defectos.

Una vez obtenida la imagen binarizada con la región de defectos se le aplica un filtro de detección de bordes para marcar una línea que rodee el defecto mostrando su ubicación. En este caso se ha utilizado el filtro por defecto de detección de bordes de ImageJ.

Observamos que el proceso de detección de bordes invierte los colores, la línea queda en blanco y el fondo en negro, así que invertimos los colores utilizando uno de los filtros de ImageJ.

Hecho esto, solo queda poner el fondo transparente y superponer la imagen del borde sobre la imagen original.

■ Multihilo

Como ya hemos dicho varias veces, uno de los objetivos del proyecto es mejorar el rendimiento de la versión previa. Una de las principales modificaciones para conseguir esto ha sido la inclusión de una estrategia multihilo, aprovechando las oportunidades que nos brindan los procesadores actuales.

Lo que hemos hecho ha sido dividir la imagen en tantas partes como procesadores disponibles tenga la máquina que está ejecutando el programa. Por ejemplo, si tenemos 2 procesadores, la imagen se dividirá en 2, en su dimensión vertical. Hay que tener en cuenta un pequeño margen, necesario para que el cálculo de algunas características sea correcto. Por ello, en el ejemplo ante-

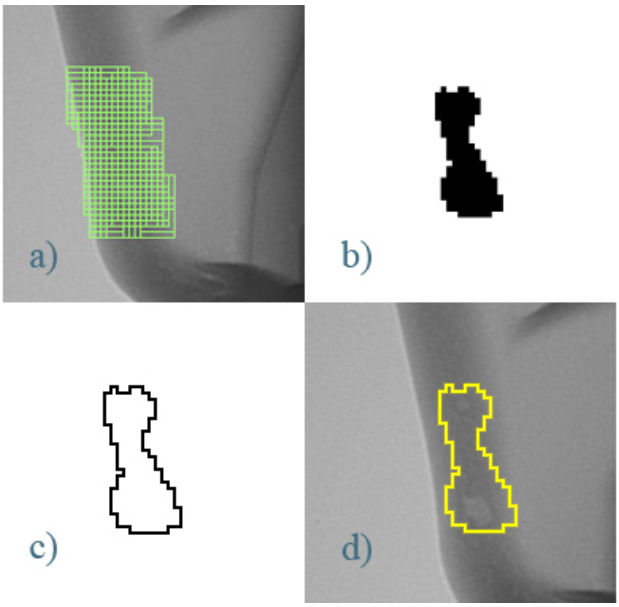

Figura 2.8.: Proceso de dibujado de defectos en nuestro proyecto.
a)Imagen con defectos marcados
b)Imagen binarizada con el área de defectos
c)Imagen con filtro de detección de bordes
d)Resultado final

rior, las 2 imágenes no serían exactamente de la mitad de altura que la original, si no que serían un poco más grandes. Hay un pequeño solapamiento.

Una vez que tenemos dividida la imagen, se ejecuta el proceso de detección o de entrenamiento sobre cada uno de los trozos de imagen. Con esto obtenemos un rendimiento mucho mayor que la versión del año pasado.

2.3.2 Fundamentos teóricos de la versión final

Con las modificaciones descritas en el apartado anterior, ya observamos que tanto el rendimiento como la precisión habían aumentado con respecto al año pasado. Aún así, teniendo en mente la posibilidad de calcular otra serie de características (como las geométricas) sobre los defectos detectados para, en un futuro, poder clasificarlos en sus diferentes tipos, se decidió intentar mejorar aún más esta precisión.

Se implementaron dos nuevas aproximaciones a la hora de detectar defectos (por lo tanto, el proceso de entrenamiento cambia). En ambas aproximaciones nos valemos de una imagen segmentada con los filtros de umbrales locales que ya hemos visto. Lo que cambia entre ellas es cómo consideramos estos filtros.

■ Primera innovación: detección normal con posterior intersección con umbrales locales

En esta primera aproximación, primero se realiza la detección de defectos como en la primera versión del proyecto. La diferencia está en el dibujado definitivo de los defectos.

Con la detección de defectos normal, obtenemos una matriz del mismo tamaño que la imagen, a partir de la cual se dibujan los defectos. En esta matriz se guarda, en cada posición (es decir, en cada píxel), el número de ventanas que los han cubierto y que se han marcado como defecto. Si este número supera cierto umbral, habrá una nueva matriz, en la que habrá un uno en todos aquellos píxeles en los que la probabilidad de ser defecto es mayor. A partir de esta matriz, se crea el dibujado definitivo.

En nuestra aproximación, introducimos un paso intermedio antes de generar la matriz definitiva. En vez de poner directamente un uno o un cero, lo que hacemos es segmentar la imagen mediante el filtro de umbrales locales. A continuación, los píxeles que superan el umbral seleccionado son comparados con la imagen de umbrales locales. Si en esta imagen aparecen como blancos, se consideran defecto. Si no, se descarta.

■ Segunda innovación: píxeles blancos en umbrales locales

Esta aproximación difiere bastante más respecto a las opciones ya vistas.

En este caso, lo primero que se hace es generar la imagen de umbrales locales. A partir de esta imagen, generamos una lista con las coordenadas de los píxeles que se han marcado como blancos.

Después, binarizamos la imagen de umbrales locales para aplicar un análisis de partículas sobre ella, con el objetivo de obtener todas las regiones que destacan, a las que consideramos regiones candidatas de albergar un defecto. Este análisis de partículas se hace con las clases de ImageJ, que va a aplicar ciertas medidas a los objetos que vaya encontrando en una imagen, buscando sus bordes [4].

Cuando tenemos la lista y las regiones, vamos sacando coordenadas de la lista y vamos determinando la región a la que pertenecen. Dependiendo del tamaño de la región y el tamaño de la ventana, se considera la coordenada o se desecha. En caso de que se considere, centramos una ventana sobre ella y aplicamos el cálculo de características y clasificación ya vistos anteriormente.

El dibujado de los defectos se realiza con la misma matriz que en la primera versión del proyecto.

La parte de analizar las regiones para determinar si hay que considerar la coordenada o no fue añadida después de una primera versión de esta opción, en la que se consideraba toda la lista de píxeles blancos. Se decidió cambiar porque si considerábamos toda la lista obteníamos muchísimos falsos positivos, con lo que la precisión no era buena.

Para aumentar el rendimiento, antes de empezar a calcular características, se divide la lista de píxeles blancos en tantas partes como procesadores disponibles haya. Después, habrá tantos hilos como partes, en los que en cada uno de ellos se iterará sobre cada una de las partes, de forma paralela.

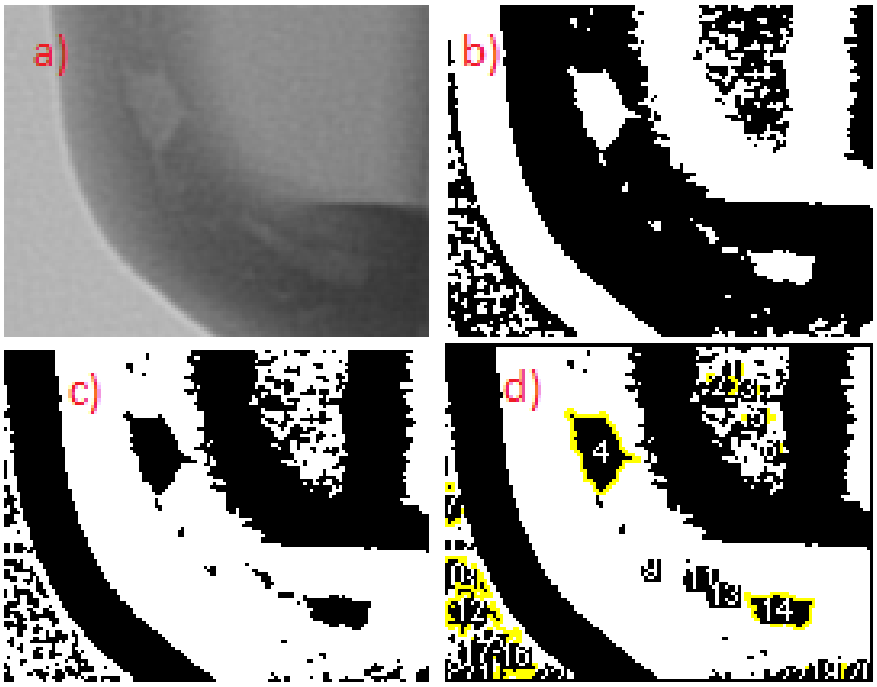

Figura 2.9.: Proceso de detección en la segunda opción
a)Imagen original
b)Imagen segmentada con los umbrales locales
c)Binarización de los umbrales locales
d)Identificación de las regiones candidatas

■ Cálculo de características geométricas

Una vez que se ha realizado el proceso de detección y dibujado de defectos, podemos calcular los descriptores geométricos de estos defectos. Para ello, se usa la imagen binarizada con la región de defectos para aplicar sobre ella un proceso de segmentación, a través del cual vamos a poder obtener una serie de regiones, sobre las cuales se puede aplicar el cálculo de las características geométricas ya mencionadas. Con estos resultados creamos una tabla que se irá refrescando si el usuario selecciona otro umbral de detección.

Estas características permitirán, en un futuro, aplicar un proceso de clasificación sobre las regiones detectadas como defecto en distintos tipos (burbuja, poros...).

■ Precision & Recall

En reconocimiento de patrones y recuperación de información, *precision* (precisión, también llamada valor predictivo positivo) es la fracción de instancias recuperadas que son relevantes, mientras que *recall* (exhaustividad, también llamada sensitividad) es la fracción de instancias relevantes que son recuperadas [54].

Para tareas de clasificación, los términos verdaderos positivos, verdaderos negativos, falsos positivos y falsos negativos comparan los resultados del clasificador con juicios externos de confianza. Los términos positivo y negativo se refieren a la predicción del clasificador (también conocida

como expectación), y los términos verdadero y falso hacen referencia a si la predicción se corresponde con el ya mencionado juicio externo (también conocido como observación). En la imagen (ver figura 2.10) se puede ver mejor cómo se ilustran estas situaciones.

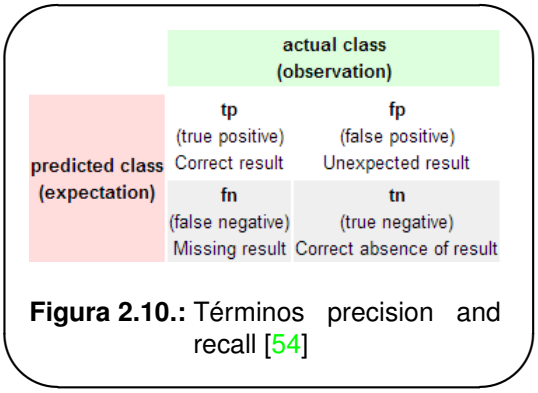

Figura 2.10.: Términos precision and recall [54]

Con esta idea en mente, ya se pueden entender las fórmulas que definen a *precision*:

$$Precision = \frac{tp}{tp + fp}$$

Y a *recall*:

$$Recall = \frac{tp}{tp + fn}$$

Con estas medidas podemos ver cómo ha sido de preciso el proceso de detección de defectos, mediante la comparación de cuántos píxeles han sido clasificados realmente como defecto y cuántos son realmente defectuosos. Esta información se saca de las máscaras que ya vimos antes.

3. TÉCNICAS Y HERRAMIENTAS

En este apartado se comentan las técnicas y herramientas utilizadas durante el desarrollo del proyecto.

3.1 Técnicas

En esta sección aparecen recogidas las técnicas utilizadas para la realización del proyecto.

3.1.1 Metodología Scrum

Scrum [11] es una metodología para la gestión y desarrollo de proyectos software basada en un proceso iterativo e incremental. Cada iteración termina con una pieza de software ejecutable que incorpora una nueva funcionalidad o mejora las ya existentes. Estas iteraciones suelen durar de dos a cuatro semanas.

Enumerando los elementos clave de *Scrum* según *Control Chaos* [8]:

- *Scrum* es un proceso ágil para gestionar y controlar el trabajo de desarrollo.
- *Scrum* es un envoltorio para prácticas de ingeniería existentes.
- *Scrum* es una aproximación basada en equipos para desarrollar sistemas y productos iterativa e incrementalmente cuando los requisitos cambian rápidamente.
- *Scrum* es un proceso que controla el caos de necesidades e intereses en conflicto.
- *Scrum* es una forma de mejorar las comunicaciones y maximizar la cooperación.
- *Scrum* es una forma de detectar y eliminar cualquier cosa que se interponga en el desarrollo y distribución de productos.
- *Scrum* es una forma de maximizar la productividad.
- *Scrum* es escalable desde un único proyecto a organizaciones completas. Ha controlado y organizado el desarrollo e implementación de muchos productos y proyectos interrelacionados con más de mil desarrolladores e implementadores.
- *Scrum* es una forma de que todo el mundo se sienta bien con su trabajo, sus aportaciones, y que ellos han hecho lo mejor que pueden hacer.

Los principales beneficios que aporta *Scrum* son:

- Entrega mensual o trimestral de resultados lo cual aporta las siguientes ventajas:
  - Gestión regular de las expectativas del cliente y basada en resultados tangibles: el cliente establece sus prioridades y cuando espera tenerlo acabado.

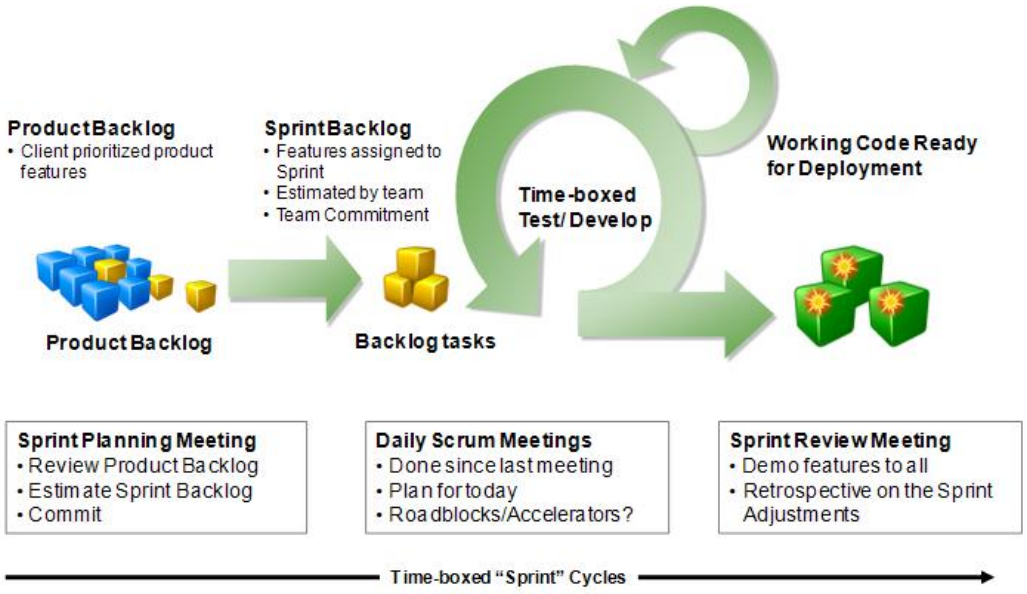

Figura 3.1.: Diagrama de la metodología Scrum [10]

- Resultados anticipados: el cliente puede empezar a utilizar los resultados mas importantes antes de que esté totalmente finalizado el proyecto.
- Flexibilidad de adaptación respecto a las necesidades del cliente, cambios en el mercado, etc.
- Gestión sistemática del retorno de inversión (*ROI*): el cliente maximiza el *ROI* del proyecto, de este modo cuando el beneficio pendiente de obtener es menor que el coste del desarrollo el cliente puede finalizar el proyecto.
- Mitigación sistemática de riesgos del proyecto: la cantidad de riesgo a la que se enfrenta el equipo está limitada a los requisitos que se puede desarrollar en una iteración.

- Productividad y calidad: de manera regular el equipo de desarrollo va mejorando y simplificando su manera de trabajar.
- Alineamiento entre cliente y el equipo de desarrollo: todos los participantes del proyecto conocen cuál es el objetivo a conseguir. El producto se enriquece con las aportaciones de todos.
- Equipo motivado: las personas están más motivadas cuando pueden usar su creatividad para resolver problemas y cuando pueden decidir organizar su trabajo.

En el diagrama (ver figura 3.1) se muestra el funcionamiento y actividades de la metodología *Scrum*.

Algunos conceptos básicos para entender *Scrum* son:

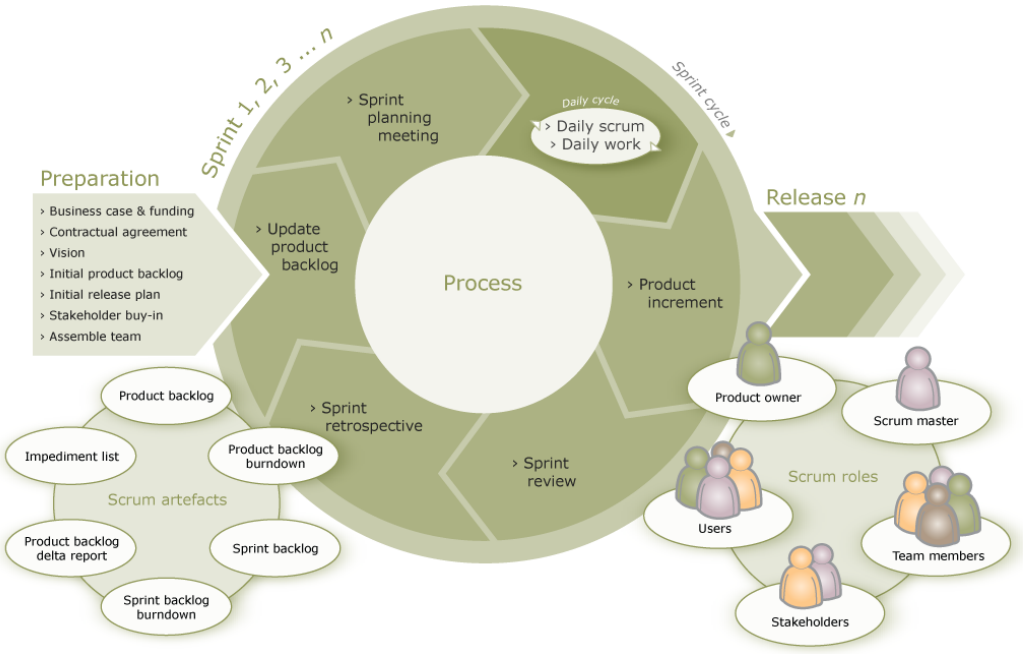

Figura 3.2.: Diagrama de etapas en Scrum [9]

- *Product Backlog*: conjunto de historias de usuario que representan los requisitos funcionales y no funcionales. Se trata de una lista priorizada en función de lo que el cliente da mayor importancia.
- *Sprint Backlog*: conjunto de tareas extraídas del *Product Backlog* y que serán realizadas durante un *Sprint*.
- *Burndown Chart*: gráfico de tareas pendientes por hacer. Representan el esfuerzo y ofrece información sobre la evolución del proyecto.

El objetivo del diagrama adjunto (ver figura 3.2) es el de agrupar y sintetizar todos los elementos de la metodología *Scrum*.

Roles

Scrum [12] define una serie de roles que se dividen en dos grupos: «gallinas» y «cerdos».

- Roles «cerdo»: son aquellos que están comprometidos a construir el software de manera regular y frecuente.
  - *Product Owner*: representa la voz del cliente. Debe asegurarse de que el equipo trabaja de forma adecuada desde la perspectiva de negocio. Escribe las historias de usuario, las prioriza y las coloca en el *Product Backlog*.
  - *Scrum Master*: su principal trabajo es eliminar obstáculos que puedan hacer que el equipo no alcance el objetivo al final del sprint, no es el líder del equipo, pero sirve de pantalla y protección.

    - *Team*: es el equipo de desarrollo y su responsabilidad es la de generar y entregar el producto (diseñadores, programadores,...).

- Roles «gallina»: en realidad no son parte del proceso *Scrum*, pero deben tenerse en cuenta. Estos roles deben participar en el proceso.

    - *Stakeholders*: agrupa a la gente que hace posible el proyecto y para quienes el proyecto producirá el beneficio que justifica el coste. Dentro de este grupo estarían los clientes, *stakeholders* ...

    - *Managers*: son la gente que establece el ambiente para el desarrollo del producto.

Reuniones

Scrum define una serie de reuniones para el correcto funcionamiento del equipo. Éstas se encuentran bien definidas en cuanto a contenido y a tiempo empleado.

- *Daily Scrum*: Cada día de un *Sprint* se realiza una reunión sobre el estado del proyecto.

    - La duración es fija (15 minutos) independientemente del tamaño del equipo.

    - La reunión debe comenzar puntualmente a la hora. A menudo hay castigos para quien lo incumple.

    - Todos pueden estar presentes pero solo pueden hablar los «cerdos».

    - La reunión se realiza de pie, facilitando no alargar la reunión.

    - El lugar y la hora deben ser fijos todos los días.

    - Preguntas que debe responder cada miembro del equipo:

        - ¿Qué has hecho desde ayer?

        - ¿Qué vas a hacer hoy?

        - ¿Qué obstáculos te has encontrado?

- *Sprint Planning Meeting*: al inicio de cada *Sprint* se debe llevar a cabo una.

    - Crear y planificar, el equipo completo, el *Sprint Backlog*. Se obtiene extrayendo tareas del *Product Backlog*.

    - Límite de ocho horas.

- *Sprint Review Meeting*: al finalizar cada *Sprint*.

    - Revisar el trabajo que fue planificado y no ha sido completado.

    - Presentar el trabajo completado a los interesados. El trabajo incompleto no puede ser mostrado.

    - Límite de cuatro horas.

- *Sprint Retrospective*: al finalizar cada *Sprint*.

    - Se analiza el *Sprint* y todos los miembros del equipo analizan qué mejoras podrían aplicarse.

- Su objetivo es la mejora continua.
- Límite de cuatro horas.

3.1.2 Java

Java es un lenguaje de programación que data de finales de los años 70. Ha tenido una gran implantación debido a la sencillez respecto a otros lenguajes orientados a objetos como *C++*.

Java ofrece un API (Application Program Interface) que ofrece a los programadores una serie de librerías y facilidades para el desarrollo de aplicaciones *Java*.

El API se encuentra divido en paquetes, que son la estructura de organización lógica. En su interior se encuentran una gran cantidad de clases que cubren un amplio abanico de funcionalidades del desarrollo software en general.

La documentación del API se encuentra disponible en la web y su consulta resulta imprescindible para cualquier tipo de desarrollo en *Java*.

■ Máquina virtual

La máquina virtual es un programa capaz de interpretar y ejecutar instrucciones expresadas en un código especial (el *Java bytecode*). Este código se obtiene al compilar el fuente original con el compilador de *Java*.

Este código es un lenguaje máquina de bajo nivel que es interpretado por la máquina virtual para realizar las operaciones. Esto hace que el rendimiento de los programas escritos en *Java* sea inferior ya que aparece una nueva pieza intermedia que es la máquina virtual.

La ventaja de ser un lenguaje interpretado es que cualquier programa escrito en *Java* puede ejecutarse en cualquier *hardware* o sistema operativo, la única condición necesaria es que exista una máquina virtual disponible.

3.1.3 UML

El Lenguaje Unificado de Modelado (UML) [18] es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad. Se trata de un lenguaje gráfico, llamado «lenguaje de modelado», que se utiliza para visualizar, especificar, construir y documentar un sistema, describiendo sus métodos o procesos. Es el lenguaje en el que está descrito el modelo.

UML permite modelar la estructura, comportamiento y arquitectura de las aplicaciones. Además, la programación orientada a objetos, que ha sido la elegida para este proyecto, es un complemento perfecto de UML. Por estas razones se ha elegido UML, en su versión 2.0.

UML cuenta con varios tipos de diagramas, los cuales muestran diferentes aspectos de las entidades representadas. Para la realización del proyecto se han utilizado los siguientes tipos:

Diagrama de Casos de Uso: muestra los casos de uso, actores y sus interrelaciones.

Diagrama de Paquetes: muestra como los elementos de modelado están organizados en paquetes, ademas de las dependencias entre esos paquetes.

Diagrama de Clases: representa una colección de elementos de modelado estáticos, tales como clases y tipos, sus contenidos y sus relaciones.

Diagrama de Secuencias: modela la lógica secuencial, ordenando en el tiempo los diferentes mensajes entre entidades.

El análisis, diseño e implementación del sistema se ha realizado empleando esta técnica, gracias a lo aprendido en las diferentes asignaturas de la carrera.

3.1.4 Weka

Weka[48] es una plataforma de software para aprendizaje automático y minería de datos, diseñada por la Universidad de Waikato. Está escrita en *Java*. Se trata de software libre distribuido bajo licencia *GNU*.

Las principales ventajas que nos ofrece son:

1. Se encuentra escrita en *Java*, lenguaje que se va a usar en el proyecto.
2. Una consecuencia de lo anterior es que *Weka* es portable, puede funcionar en cualquier sistema operativo.
3. Se dispone del código fuente para ver cómo hace las cosas y cómo funciona.
4. Buena documentación (*JavaDoc*) para ayudar a la programación.
5. Completa API capaz de representar de una manera sencilla la abstracción de las instancias.
6. Es capaz de obtener los datos de diversos orígenes, tanto de texto como *ARFF* y *CSV*, como de bases de datos.
7. Su uso está muy generalizado en el mundo de la minería de datos.
8. Ya utilizado con anterioridad, por lo que su manejo no resulta nuevo.

No todo son ventajas y a continuación se detallan los inconvenientes valorados:

1. *Weka* carga todas las instancias en memoria por lo que se limita el número de instancias que es capaz de manejar.

Weka soporta varias tareas estándar de minería de datos, especialmente, preprocesamiento de datos, clustering, clasificación, regresión, visualización, y selección. Todas las técnicas de *Weka* se fundamentan en la asunción de que los datos están disponibles en un fichero plano (flat file) o una relación, en la que cada registro de datos está descrito por un número fijo de atributos (normalmente numéricos o nominales, aunque también se soportan otros tipos).

3.1.5 ImageJ

ImageJ [7] es un programa de procesamiento de imagen digital de dominio público programado en *Java* desarrollado en el National Institute of Health.

La licencia de ImageJ es la siguiente:

> ImageJ is a work of the United States Government. It is in the public domain and open source. There is no copyright. You are free to do anything you want with this source but I like to get credit for my work and I would like you to offer your changes to me so I can possibly add them to the, "official" version.

Lo que quiere decir básicamente que somos libres de hacer lo que queramos con ImageJ pero que si realizamos algún cambio deberíamos ofrecérselos al creador para que los añada a la versión «oficial».

ImageJ fue diseñado con una arquitectura abierta que proporciona extensibilidad vía plugins *Java* y macros (macroinstrucciones) grabables. Se pueden desarrollar plugins de escaneo personalizado, análisis y procesamiento usando el editor incluido en ImageJ y un compilador *Java*. Los plug-ins escritos por usuarios hacen posible resolver muchos problemas de procesado y análisis de imágenes, desde de imágenes en vivo de las células en tres dimensiones, procesado de imágenes radiológicas, comparaciones de múltiples datos de sistema de imagen hasta sistemas automáticos de hematología.

Aunque ImageJ es extensible mediante plugins y macros, nosotros lo hemos elegido con la finalidad de utilizarlo como librería.

ImageJ puede mostrar, editar, analizar, procesar, guardar, e imprimir imágenes de 8 bits (256 colores), 16 bits (miles de colores) y 32 bits (millones de colores). Puede leer varios formatos de imagen incluyendo TIFF, PNG, GIF, JPEG, BMP, DICOM, FITS, así como formatos RAW (formato plano).

ImageJ soporta pilas o lotes, una serie de imágenes que comparten una sola ventana, y es multiproceso, de forma que las operaciones que requieren mucho tiempo se pueden realizar en paralelo en hardware multi-CPU.

ImageJ puede calcular el área y las estadísticas de valor de píxel de selecciones definidas por el usuario y la intensidad de objetos umbral (thresholded objects). Puede medir distancias y ángulos. Se puede crear histogramas de densidad y gráficos de línea de perfil.

Es compatible con las funciones estándar de procesamiento de imágenes tales como operaciones lógicas y aritméticas entre imágenes, manipulación de contraste, convolución, análisis de Fourier, nitidez, suavizado, detección de bordes y filtrado de mediana. Hace transformaciones geométricas como ampliar, rotación y simetrías. El programa es compatible con cualquier número de imágenes al mismo tiempo, limitado solamente por la memoria disponible.

Preferimos usar ImageJ frente a otras opciones, como OpenCV, principalmente por su buena documentación (*JavaDoc*), cosa que facilita mucho la programación.

3.1.6 Patrones de diseño

Un patrón de diseño es una solución a un problema de diseño común en el desarrollo de software.

La principal característica de un patrón de diseño es que debe ser reusable, es decir, debe poder aplicarse a diferentes problemas de diseño en distintas circunstancias y ser efectivo.

Los patrones de diseño utilizados para el desarrollo de este proyecto se explican en el anexo 3 de esta memoria.

3.2 Herramientas

En esta sección aparecen cada una de las herramientas utilizadas para la realización del proyecto.

3.2.1 Eclipse

Como entorno de desarrollo de la aplicación se utilizará Eclipse. La decisión fue tomada por haber trabajado anteriormente con esta herramienta y conocer sus ventajas e inconvenientes. La disponibilidad de *plugins* disponibles facilita el desarrollo, integrando el control de versiones, *suites* de pruebas, ...

Eclipse es un producto realizado por la *Eclipse Foundation*, que es una comunidad de codigo abierto que tiene como objetivo desarrollar una plataforma para el desarrollo software.

Se encuentra escrito en *Java* bajo una licencia propia, la *EPL* (Eclipse Public License [31]).

Aunque en su origen se creó para *Java* existen versiones de todo tipo para otros lenguajes como pueden ser *C* o adaptaciones comerciales para productos o lenguajes concretos.

Página web de la herramienta: `http://www.eclipse.org/`.

3.2.2 JUnit

JUnit es un conjunto de bibliotecas o *framework* que son utilizadas para realizar las pruebas unitarias de aplicaciones *Java*. Dispone de una buena reputación dentro de la literatura sobre programación de pruebas.

El propio *framework* permite visualizar los resultados en texto, como gráficos o como tarea de *Ant*.

Se utilizará el *plugin* de Eclipse por la facilidad que aporta para la ejecución de las pruebas y su total integración con el entorno de desarrollo.

Se ha utilizado por haber sido utilizado durante la carrera y ser el lanzador más conocido por los desarrolladores.

Página web de la herramienta: `http://www.junit.org/`.

3.2.3 JDepend

Se trata de una herramienta de métricas que permite conocer información de utilidad de un proyecto software.

Se utilizará el *plugin* para Eclipse que permite visualizar los valores de las métricas y la gráfica comparativa de cada paquete.

Página web de la herramienta: `http://www.clarkware.com/software/JDepend.html`.

3.2.4 Source Monitor

Se trata de una herramienta de análisis de código capaz de analizar proyectos escritos en diversos lenguajes.

Su utilización ha sido motivada porque permitirá comparar las métricas obtenidas con los umbrales establecidos por la Universidad de Burgos.

Página web de la herramienta: http://www.campwoodsw.com/sourcemonitor.html.

3.2.5 Astah

Es una herramienta de modelado *UML* creado por la compañía *ChangeVision*.

Al estar pensado para *Java*, permite la importación y exportación de código fuente y la generación de gráficos automáticos.

En el proyecto se utilizará la versión *Community* porque es gratuita y cubre las necesidades de modelado del proyecto.

Página web de la herramienta: http://astah.net/editions/community.

3.2.6 GitHub

GitHub es una plataforma para el desarrollo colaborativo de software que utiliza el control de versiones *Git*.

Es una herramienta completa y robusta que permite la creación de grupos, ramas, etiquetas y todo tipo de artefactos necesarios para la organización de un proyecto de programación.

Las principales ventajas que nos ofrece son:

- Nuestro código queda alojado en la nube, permitiendo acceder a él desde cualquier lugar.
- Nos permite trabajar con la metodología de *Rama por tarea*, en la que se crea una rama por cada tarea a realizar, permitiendo trabajar en dos tareas simultáneamente, fusionando después los cambios.
- Tiene cliente propio multipltaforma, lo que nos permite gestionar el repositorio de una forma muy sencilla, pudiendo validar los cambios que hagamos en el código y subiendo estos cambios al servidor.

Página web de la herramienta: https://github.com/.

3.2.7 PivotalTracker

PivotalTracker será utilizada como herramienta de gestión y control de tareas y errores.

Está especializada en proyectos ágiles por lo que da soporte a todos los conceptos de la metodología *Scrum* utilizada en el proyecto.

Se utilizará la versión de *hosting* que permite acceder desde cualquier equipo al servidor desde un navegador. Además, permite sincronizarse con *GitHub*, con lo que podemos ver cómo se van realizando las tareas tanto en la propia herramienta como en el código. En el Apéndice A, aparece un breve manual de usuario que permite visualizar la planificación y las tareas.

Página web de la herramienta: https://www.pivotaltracker.com/.

3.2.8 MiKTEX

MikTEX es una distribución TEX/LATEX libre y de código abierto para Windows.

Una de sus características es la capacidad que tiene para instalar paquetes automáticamente sin necesidad de intervención del usuario. Al contrario que otras distribuciones, su instalación es extremadamente sencilla.

Ha sido utilizada por recomendación de uno de los tutores quien suministró un tutorial sobre su instalación.

Página web de la herramienta: http://miktex.org/.

3.2.9 TEXMaker

TEXMaker es un editor de LATEX multiplataforma similar a *Kile*.

Aunque existen multitud de editores para LATEX se ha escogido este por haber sido recomendado por uno de los tutores, dado que nunca antes se ha trabajado con este lenguaje de documentación se aceptó la sugerencia. Además, aportó un tutorial de como instalar y configurar el editor en el sistema operativo Windows.

Una de sus características es la posibilidad de trazabilidad de código desde PDF. Configurando el editor y un visor de PDF, el programa es capaz de detectar la línea a la que se corresponde un determinado comando LATEX. Esto es especialmente ventajoso cuando, como en este caso, no se conoce el lenguaje.

En el enlace http://en.wikipedia.org/wiki/Comparison_of_TeX_editors aparece una comparativa entre diversos editores disponibles.

Página web de la herramienta: http://www.xm1math.net/texmaker/.

3.2.10 WindowBuilder

WindowBuilder es un *plugin* de Eclipse que permite diseñar de una forma fácil y rápida interfaces gráficas basadas en *Swing*.

Hemos elegido este editor de interfaces gráficas debido a su facilidad para crearlas, ya que incluye un editor *WYSIWYG*, con el que podemos arrastrar los elementos a la ventana que estamos creando y moverlos hasta dejarlos en la posición deseada.

La otra parte interesante de este *plugin* es que genera el código automáticamente, con lo que nos ahorra mucho trabajo. Además, el código que genera está bien agrupado, con lo que después es muy fácil refactorizarlo.

Página web del *plugin*: http://www.eclipse.org/windowbuilder/.

3.2.11 Auto Local Threshold

Auto Local Threshold es un *plugin* de ImageJ que implementa la segmentación de imágenes mediante los filtros de umbrales locales.

Por defecto, ImageJ no tiene esta funcionalidad, así que tuvimos que descargarnos e instalar este *plugin*, que permite realizar estas opciones de forma muy sencilla.

Para poder usarlo en el proyecto, tuvimos que cambiar alguna cosa de la clase del *plugin* para poder usarlo con nuestro código.

Página web del *plugin*: http://fiji.sc/wiki/index.php/Auto_Local_Threshold.

3.2.12 Apache Commons IO

Apache Commons IO es una librería de utilidades para asistir al desarrollo de funcionalidad relacionada con entrada/salida.

Hemos decidido utlizarla debido a que simplifica mucho el realizar algunas operaciones con ficheros, como es la fusión de uno o más ficheros de texto, o la exportación de un cierto texto a un fichero externo.

La página web de la herramienta es: http://commons.apache.org/proper/commons-io/.

3.2.13 EJML

EJML (Efficient Java Matrix Library) es una librería de código abierto de álgebra lineal para manipular matrices densas. Uno de los objetivos que se marcaron sus desarrolladores fue que sea lo más eficiente posible desde un punto de vista computacional y espacial, ya sea al manipular matrices grandes o pequeñas.

Tiene una buena API que permite realizar un gran número de operaciones, desde las más sencillas, como sumar o multiplicar, hasta las más complicadas, como descomposiciones de diversos tipos.

Se distribuye con una licencia LGPL. Puede encontrarse en https://code.google.com/p/efficient-java-matrix-library/.

3.2.14 Zotero

A la hora de realizar cualquier trabajo de investigación o documento de cierta envergadura, como puede ser la memoria de un trabajo final de carrera, el autor del mismo se va a encontrar con un gran número de información a su disposición. Toda esta información proviene de muchas y muy diversas fuentes (libros, artículos, revistas, bases de datos, internet...) y tiende a crecer de forma incontrolada a lo largo del desarrollo del proyecto. Es por ésto por lo que surge la necesidad de utilizar algún sistema que gestione toda esta información de forma eficaz.

Este sistema es un gestor bibliográfico, con el que además de gestionar esta información, también se puede insertar citas en los documentos y crear la bibliografía en un formato normalizado.

Zotero es un gestor bibliográfico de código abierto que se instala como una extensión del navegador Firefox y del navegador Chrome. De esta forma, permite añadir directamente a la bibliografía las páginas web visitadas y buscar documentos en línea.

Permite exportar la bibliografía en formato LaTeX directamente, lo cual es muy cómodo para integrarla directamente en el documento.

Página web de la herramienta: http://www.zotero.org/.

4. ASPECTOS RELEVANTES DEL DESARROLLO DEL PROYECTO

En este apartado se detallan los aspectos más relevantes que se han encontrado en el proceso de desarrollo del proyecto.

4.1 Problemas y retos

El tema principal del proyecto era totalmente desconocido. Para la obtención de los defectos en las radiografías se han utilizado técnicas novedosas que no se enseñan en ninguna asignatura de la carrera. Hay que añadir que algunas de esas técnicas se usan actualmente en proyectos de investigación en universidades y centros de investigación con mucha más experiencia en este campo que la Universidad de Burgos.

Se busca resolver un problema real, con imágenes reales (cedidas por el Grupo Antolín) obtenidas desde los puntos de vista y condiciones que obtiene una máquina real, piezas reales y con formas muy complejas. Es un problema mucho más complicado que el que se aborda en los artículos relacionados donde la imagen suele ser mucho mas uniforme y sencilla. En la imagen (ver figura 4.1) se puede ver una comparativa de algunas de las imágenes usadas en otros artículos. En la figura (ver figura 4.2) se puede ver un ejemplo de una de las imágenes que usamos en nuestro proyecto.

Por lo anterior, este ha sido un proyecto de gran incertidumbre y riesgo que ha hecho que:

1. Como se preveía que los requisitos del proyecto iban a cambiar mucho durante el desarrollo del mismo, ya que surgen nuevas ideas, nuevas aproximaciones para resolver el problema, se decidió usar la metodología Scrum, que está pensada para este tipo de entornos, en los que los requisitos cambian y es difícil hacer una planificación.
2. Haya sido necesario adquirir muchos nuevos conocimientos.

4.2 Mejoras respecto a la versión previa

Uno de los objetivos del proyecto era mejorar a la versión previa presentada el curso pasado. Esto supuso algunos problemas, ya que arreglar ciertos aspectos no ha sido tan sencillo.

Las mejoras introducidas han sido:

1. **Mejoras sobre el código:** se ha mejorado el código en general, procurando que sea más claro y evitando ciertos defectos de código, en la medida de lo posible. Se han refactorizado, por ejemplo, algunas clases en las que había métodos exageradamente largos, procurando dividirlos en métodos más pequeños, lo que mejora mucho la legibilidad.

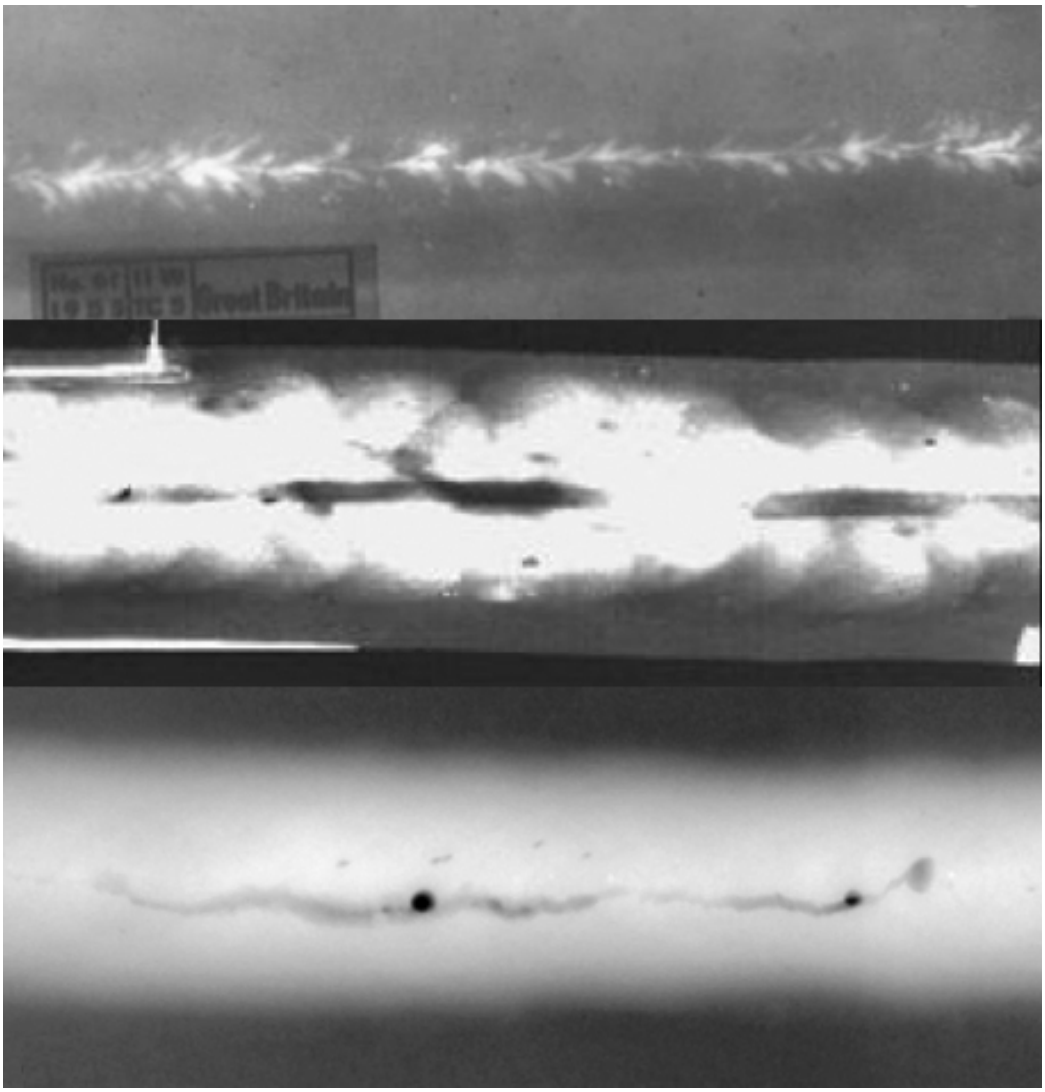

Figura 4.1.: Radiografías usadas en otros artículos [13] [33] [35]

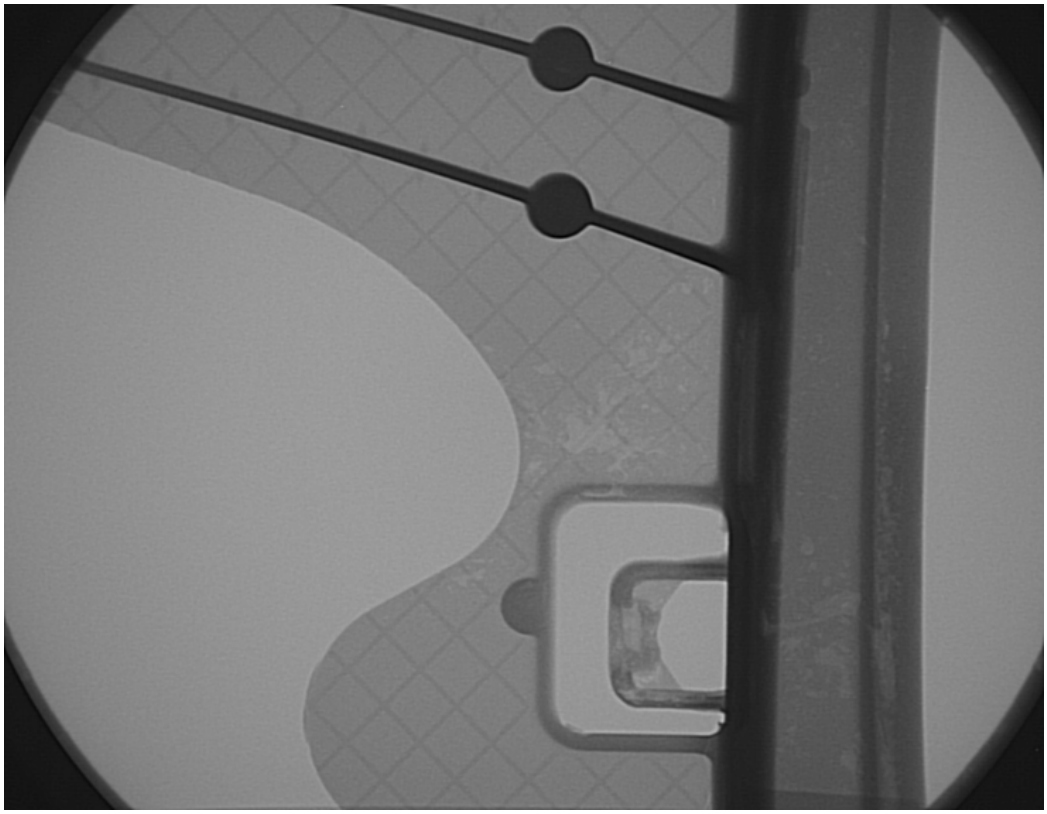

Figura 4.2.: Ejemplo de radiografía usada en nuestro proyecto

2. **Mejoras en la estructura:** se ha cambiado completamente la estructura de la aplicación, introduciendo un diseño en tres capas y algunos patrones de diseño que permiten mejorar el mantenimiento y la extensibilidad de la aplicación.
3. **Mejoras de rendimiento:** se ha intentado mejorar el rendimiento general de la aplicación. Básicamente, esto lo hemos conseguido mediante el proceso paralelo de los distintos trozos de la imagen mediante multihilo, aunque también se han cambiado algunos cálculos para que sean más eficientes.
4. **Mejoras en la interfaz:** la interfaz se cambió casi completamente, buscando una mejor intuitividad para el usuario. Por ejemplo, los botones son ahora más claros y se desactivan cuando no se pueden usar, cosa que antes no pasaba y podía llegar a causar problemas.
5. **Mejoras en la documentación y ayuda:** se ha mejorado la calidad de la API, extendiéndola a todos los elementos del código y arreglando fallos del lenguaje. Además, se ha añadido un módulo de ayuda en línea para permitir al usuario consultar cualquier duda de una forma rápida y sencilla [AÚN NO ESTÁ HECHO].
6. **Mejoras en la precisión:** se ha buscado mejorar la detección de defectos implementando nuevas aproximaciones y mejorando la que ya existía.
7. **Ampliación de funcionalidad:** no sólo se han añadido nuevas aproximaciones para la detección de defectos, sino que también se ha ampliado la funcionalidad de la aplicación en otras formas, como por ejemplo, el cálculo de características geométricas, la interactividad con los defectos detectados, las medidas de precision & recall,...

En definitiva, se ha intentado que, a partir de la buena base que representaba el proyecto del año pasado, se pueda ampliar esta idea de una forma mucho más sencilla. Se ha buscado, por tanto, crear una aplicación mucho más fácil de ampliar, ya que este proyecto es candidato a recibir una innumerable cantidad de mejoras prometedoras en un futuro.

4.3 Conocimientos adquiridos

Durante el desarrollo del proyecto se han ido aprendiendo y perfeccionando distintas disciplinas, las cuales aparecen detalladas a continuación.

4.3.1 Minería de datos

Al inicio del proyecto el conocimiento sobre la minería de datos estaba limitado a los conocimientos adquiridos en la asignatura de Minería de Datos de 5º curso de Ingeniería Informática.

Por este motivo cuando se expuso el proyecto se entendió como un reto y una manera de poder aprender sobre esta interesante rama de la informática en la que entran en juego grandes volúmenes de datos.

De este modo y a base de leer artículos, se han asimilado y refinado multitud de conceptos y técnicas.

4.3.2 Weka

En la asignatura de Minería de Datos, de la que ya hemos hablado en el apartado anterior, se utilizó *Weka* para realizar las prácticas. Es por ello que ya teníamos algunos conocimientos sobre esta herramienta.

Este proyecto nos ha permitido usar *Weka* de una forma que no habíamos considerado hasta ahora, y es incluir alguno de sus métodos dentro de nuestra propia aplicación, aprovechando las posibilidades que nos brinda la herramienta.

4.3.3 Metodología Scrum

Las metodologías ágiles han adquirido un gran éxito dentro del desarrollo de software. Por este motivo el proyecto de final de carrera se presentaba como una buena base sobre la que aplicar una de estas metodologías y aprender de ella.

Se eligió *Scrum* por el hecho de que está pensado para entornos en los que cambian los requisitos y es difícil hacer una buena planificación.

Además, *Scrum* se ha explicado en la asignatura de Planificación y Gestión de Proyectos de 4º curso de Ingenería Informática. De este modo la realización del proyecto bajo esta nueva metodología ha aportado una experiencia adicional a los desarrollos clásicos en cascada que son utilizados en multitud de empresas.

4.3.4 Programación multihilo

Para poder mejorar el rendimiento de la aplicación, se hizo necesaria la programación multihilo. Ya poseíamos algunos conocimientos de algunas asignaturas de la carrera, pero nunca lo habíamos usado en Java. Por ello, ha sido necesario leer documentación al respecto y lidiar con algunos problemas que pueden presentar este tipo de aplicaciones.

4.3.5 Documentación en LaTeX

Al principio, la temida documentación se iba a desarrollar con uno de los clásicos compositores de texto, pero pronto nuestros tutores nos recomendaron utilizar encarecidamente la herramienta LaTeX [25]. Este hecho, que a priori parecía un reto sencillo, se convirtió en un proceso de cierta complejidad y con una curva de aprendizaje larga e intensa.

En estos momentos damos gracias a nuestros tutores por empeñarse en convencernos a utilizar LaTeX, ya no solo por el resultado estético que se obtiene, sino por darnos un nuevo reto a superar que añade valor a la realización de este proyecto y la documentación, además del valor para el futuro laboral.

Hemos utilizado una plantilla creada por el alumno Álvar Arnáiz González en el proyecto «Biblioteca de algoritmos de selección de instancias y aplicación orientada a su docencia» [32]. Futuros alumnos podrán disfrutarla y mejorarla. Tarde o temprano, el hecho de que cada año nuevo alumnos utilicen y mejoren esta plantilla, hará que la Universidad de Burgos tenga una plantilla estándar para la realización de cualquier memoria escrita en LaTeX.

5. TRABAJOS RELACIONADOS

En este apartado se hablará de algunos trabajos relacionados con los temas que se tratan en el proyecto.

5.1 Artículos Estudiados

A la hora de comenzar el proyecto, necesitamos leer el artículo principal sobre el que estaba basado el proyecto anterior. Además, durante el desarrollo del mismo, fue necesario leer otros artículos para buscar nueva información, como por ejemplo, cómo clasificar defectos a través de características geométricas. A continuación se muestra un listado de todos los artículos, junto con una tabla comparativa (ver tabla 5.1) de aquellos artículos que nos han parecido más interesantes. Para ver un resumen más detallado de estos artículos, se puede ir a la sección 5.2 de la memoria.

- *Automated detection of welding defects without segmentation* [35]
- *An automatic system of classification of weld defects in radiographic images* [46]
- *Recognition of welding defects in radiographic images by using support vector machine classifier* [47]
- *Image thresholding based on the EM algorithm and the generalized Gaussian distribution* [14]
- *Weld defect classification using EM algorithm for Gaussian mixture model* [33]
- *Multiclass defect detection and classification in weld radiographic images using geometric and texture features*[45]

Título	Autores	Año	Preprocesamiento	Características	Clasificador
An automatic system of classification of weld defects in radiographic images	Rafael Vilar et al.	2009	Filtro adaptativo de Wiener de 7×7. Filtro gausiano de paso bajo de 3×3. Método de Otsu.	Área, centroide, eje mayor, eje menor, excentricidad, etc.	Red neuronal artificial (ANN)

continúa en la página siguiente

continúa desde la página anterior

Título	Autores	Año	Preprocesamiento	Características	Clasificador
Weld defect classification using EM algorithm for Gaussian mixture model	M.Tridi et al.	2005	Segmentación de la imagen	Geométricas: área, longitud, anchura, elongación, perímetro, etc.	k-medias
Recognition of welding Defects in radiographic images by using support vector machine	X.Wang et al.	2010	Binarización adaptativa basada en wavelet Ecualización adaptativa del histograma	Características de Haralick, de Gabor, de matriz de co-ocurrencia y morfológicas	Máquina de vector de soporte (SVM)
Automated detection of welding defects without segmentation	Domingo Mery	2011	Saliency Map	Características estándar, de Haralick y LBP	Máquina de vector de soporte (SVM)
Multiclass defect detection and classification in weld radiographic images using geometric and texture features	Ioannis Valavanis et al.	2010	Umbrales locales de Savuola. Método de segmentación basado en grafos.	De texturas (2º momento angular, contraste, correlación, suma de cuadrados, etc) y geométricas (posición, ratio de aspecto, área, longitud, redondez...).	SVM y red neuronal artificial

Tabla 5.1.: Tabla comparativa de artículos

5.2 Revisión del estado del arte

En este apartado se examina brevemente el estado del arte relevante para el tema de este proyecto. Es importante realizar una revisión bibliográfica ya que nos permitirá saber si el problema que nos planteamos está ya resuelto, así como conocer lo que otros investigadores han aportado en nuestra línea de trabajo y cómo han planteado y realizado sus investigaciones. Además, es necesario conocer con detalle las técnicas experimentales que otros han usado en problemas parecidos al nuestro, para seguirlas o para modificarlas.

A continuación, se incluyen resúmenes de aquellos artículos que nos han parecido más interesantes para ayudarnos en el desarrollo del proyecto.

5.2.1 An automatic system of classification of weld defects in radiographic images - Rafael Vilar et al.

En este artículo [46] se estudia la forma de detectar defectos de soldadura en radiografías. Se utilizan imágenes de 8 bits con una resolución de 2900×1950 píxeles. Consta de las siguientes fases:

■ Preprocesado de las imágenes

Para reducir el ruido se utilizan dos técnicas:

- Filtro adaptativo de Wiener de 7×7 [49].
- Filtro gausiano de paso bajo de 3×3.

También se utilizan técnicas para mejorar el contraste. Finalmente, la imagen se divide en bandas de 640×480 píxeles.

■ Segmentación de las regiones de soldadura

El objetivo de esta fase es aislar la región de soldadura del resto de elementos. El proceso se desarrolla en tres fases. En la primera se busca un umbral óptimo que permite binarizar la imagen, separando los píxeles de los objetos de los píxeles del fondo. Para ello se utiliza el método de Otsu [39]. En la segunda se etiquetan los componentes conectados de la imagen binarizada. Se utiliza el procedimiento propuesto por Haralick y Shapiro [20], que devuelve una matriz con el mismo tamaño que la imagen. Los píxeles etiquetados como «0» son el fondo, los píxeles etiquetados como «1» representan un objeto, los píxeles etiquetados como «2» representan un segundo objeto y así sucesivamente. Para concluir, en la tercera fase, como un criterio para seleccionar entre los objetos etiquetados, el área máxima es establecida. De esta manera, se identifica la región de soldadura de entre todos los objetos de la imagen.

■ Segmentación de heterogeneidades

Se toma como entrada la imagen producida por la fase anterior. La salida obtenida es una imagen que contiene únicamente defectos potenciales. Primero, se binariza la imagen utilizando el método de Otsu para obtener el umbral óptimo. Después, se traza el borde exterior de los objetos. Una vez

que se ha hecho esto, se deduce que los defectos son objetos situados dentro de una región de soldadura.

■ Extracción de características

La salida de esta fase es una descripción de cada defecto candidato de la imagen. Las características extraídas son: área, centroide (coordenadas X e Y), eje mayor, eje menor, excentricidad, orientación, número de Euler [18], diámetro equivalente, solidez, extensión y posición. Se genera un vector de entrada (12 componentes) para cada defecto candidato y expertos humanos en defectos de soldadura producen un vector objetivo asociado.

■ Análisis de componentes principales

En esta fase se reduce el tamaño de los vectores de características de entrada. Para ello se utiliza la técnica PCA [40].

■ Predicción utilizando una red neuronal multicapa

Se utiliza una red neuronal multicapa para clasificar los defectos. Se implementan clasificadores de patrones no lineales de tipo supervisado utilizando ANN.

Con los datos de entrenamiento, el error es pequeño, pero cuando se introducen nuevos datos a la red el error es grande. La red ha memorizado los ejemplos de entrenamiento pero no ha aprendido a generalizar en nuevas situaciones. Para mejorar la generalización se utilizan tres técnicas:

1. Regularización.
2. Regularización de Bayes.
3. Early stopping o bootstrap.

Para evaluar el rendimiento de la red, se realiza un análisis de regresión entre la respuesta de la red y los objetivos correspondientes. El coeficiente de correlación obtenido entre las salidas y los objetivos es una medida de cómo de bien es explicada la variación de la salida por los objetivos. Si este número es igual a uno, entonces hay correlación perfecta entre los objetivos y las salidas. Para determinar el coeficiente de correlación se emplea una regresión lineal usando el método de mínimos cuadrados.

5.2.2 Weld defect classification using EM algorithm for Gaussian mixture model - M.Tridi et al.

En este articulo [33], se proponen dos algoritmos de clasificación de los defectos de soldadura *(El algoritmo Fuzzy-C-Means Iterativo: FCMI y el algoritmo Expectation maximization: EM)*. El primer algoritmo se basa en el concepto de distancia y lógica difusa, y el segundo está basado en conceptos estadísticos.

■ Algoritmo Fuzzy-C-Means Iterativo

El algoritmo *Fuzzy-C-Means Iterativo* [23] utiliza el concepto de lógica difusa y distancia para la clasificación. Está dada por el siguiente algoritmo:

1. Los centros de los clusters son inicializados en un conjunto de ejemplos
2. Cálculo de la distancia euclidiana entre cada muestra y cada centro de cluster
3. Cálculo de la función de pertenencia (Fuzzification)

■ The EM algorithm

El algoritmo *Expectation Maximization (EM)* [57], es una extensa clase de algoritmos iterativos usada para estimación de máxima verosimilitud o máxima probabilidad a posteriori en problemas en los que faltan datos.

■ Aplicación en la clasificación de defectos de soldadura

En esta aplicación se ha tomado una base de datos formada por 72 radiografías con defectos. Para poder clasificar un patrón (imagen segmentada), es esencial caracterizarlas por un vector de características. La elección de este vector está basada en el conocimiento obtenido por un experto en radiografías. Se pueden encontrar varios tipos de características, como por ejemplo: momentos Zernik, momentos Legendre, momentos Geométricos, coeficientes de Fourier etc. Las características usadas en esta aplicación son parámetros o características geométricas. Este tipo de características consisten en caracterizar un objeto acorde al vector cuyos elementos son característicos, como por el perímetro, superficie, dirección principal de la inercia inercia y elongación.

Se advierte el hecho de que los centros de los clusters representan eficientemente las cuatro clases (Y1 para roturas, Y2 para falta de penetración, Y3 para inclusión de gas e Y4 para inclusión de óxido), y son diferentes entre sí.

■ Conclusión

Se describe un nuevo enfoque para clasificar el defecto de soldadura para las imágenes de radiografía usando el algoritmo EM. El algoritmo EM es muy sensible a la elección de los valores iniciales de los parámetros. En este caso, se ha utilizado el algoritmo de k-medias para la inicialización. La principal contribución es una comparación entre los algoritmos EM y FCMI. Los resultados experimentales indican que este algoritmo ha dado mejores resultados que el algoritmo FCMI.

5.2.3 Recognition of Welding Defects in Radiographic Images by Using Support Vector Machine - X.Wang et al.

En este artículo [47] se describe un método para detectar defectos en imágenes de rayos X basado en Support Vector Machine (SVM). El método está compuesto por tres fases:

■ Preprocesado de las imágenes

Las imágenes que se van a analizar tienen bajo contraste, mucho ruido y fondo no uniforme. Para mejorar estas condiciones se utilizan dos métodos: *binarización adaptativa basada en wavelet y ecualización adaptativa del histograma*. Después, se segmenta la imagen utilizando *umbralización multi-nivel basada en entropía máxima borrosa*. Se segmentan los defectos hipotéticos (algunos son falsas alarmas).

■ Extracción de características

Este apartado se centra en la medición de las propiedades de las regiones. Se extraen dos tipos de características:

1. Características de textura Se extraen la matriz de co-ocurrencia y los filtros de Gabor [16]. Para medir estas características, se utilizan 4 de las 14 medidas propuestas por Haralick [19]:
   - Shannon Entropy
   - Contrast
   - Angular Second Moment
   - Inverse Difference Moment

   Se extraen 64 características de Gabor y 16 a partir de la matriz de co-ocurrencia.
2. Características morfológicas: área, longitud, anchura, elongación, orientación, ratio entre la anchura y el área (RWA), compacidad.

   Estas características, sumadas a las de textura, nos dan un total de 87 características

■ Clasificación de patrones

Se dividen las imágenes en regiones específicas de acuerdo a las características extraídas, clasificándolas en dos grupos («defecto» o «no defecto»). Se seleccionan 16 características combinando las 12 mejores obtenidas mediante la aplicación de un algoritmo basado en SVM y las 12 mejores obtenidas con un análisis ROC. Se entrena la SVM con los vectores formados por esas características. Después, se preprocesa la imagen de prueba para extraer las 16 características, y entonces se aplica la SVM entrenado para que decida entre «defecto» y «no defecto».

5.2.4 Multiclass defect detection and classification in weld radiographic images using geometric and texture features - Ioannis Valavanis

■ Local thresholding

Los autores usan un filtro de umbrales locales, llamado Sauvola [42], como primer paso para la detección de defectos. Los autores dicen que se comporta mejor que otros métodos, como *Otsu* o *Niblack*. Después de usar este método, se hace necesario utilizar unas operaciones morfológicas para eliminar una serie de puntos aislados (ruido). Con esto, se obtienen una serie de regiones candidatas a albergar defectos, marcadas en blanco.

■ Segmentación

El segundo paso para la detección de defectos es usar una operación de segmentación. Se utiliza un método basado en grafos que es capaz de capturar regiones perceptualmente distintas, aunque su interior se caracterice por una alta variabilidad, considerando características globales de la imagen.

■ Clasificación

Para realizar la clasificación de los defectos se hace necesario calcular una serie de descriptores de regiones, divididos en descriptores de texturas y descriptores geométricos. Los descriptores de texturas son los mismos que hemos usado en nuestro proyecto, ya que este artículo también está basado en el artículo de Domingo Mery. Entre los geométricos se encuentran el área, la redondez, semieje mayor y menor de la mayor elipse,...

Como clasificadores, los autores usan support vector machine y una red neuronal artificial, realizando una comparativa entre ambos.

6. CONCLUSIONES Y LÍNEAS DE TRABAJO FUTURAS

En este capítulo se van a exponer las conclusiones obtenidas tras el desarrollo del proyecto y las posibles líneas de trabajo futuras.

6.1 Aspectos que han complicado la realización del proyecto

Los retos más importantes que han surgido durante el proyecto y han complicado la realización del mismo han sido los siguientes:

- LaTeX: la curva de aprendizaje de este lenguaje es bastante dura y esto ha sido especialmente visible en las primeras fases del proyecto.
- Desconocimiento del *background* teórico: los conocimientos previos sobre visión artificial eran muy escasos, por lo que ha sido complicado adaptarse a tantos conocimientos nuevos.
- Programación multihilo: nos ha dado numerosos quebraderos de cabeza implementar los hilos y aplicarlos al análisis de imágenes paralelamente.
- Comprensión del proyecto del año pasado: entender el proyecto del año pasado ha sido difícil. Por una parte, por lo que ya hemos dicho sobre el desconocimiento del *background* teórico. Por otra parte, porque el código era bastante complicado de entender, no sólo por su complejidad, sino también por la falta de un diseño robusto que facilite el mantenimiento y porque la documentación no siempre era todo lo buena que esperábamos.
- Dificultades para planificar: no siempre era sencillo planificar un *Sprint*, ya que hemos tenido una carga de trabajo muy importante a lo largo de todo el curso.

6.2 Conclusiones

Las conclusiones extraídas tras el desarrollo del proyecto son detalladas a continuación:

- Se ha mejorado la precisión de la herramienta del año pasado mediante la mejora de aspectos como la determinación de cuándo una ventana es defectuosa o no, o la inclusión de nuevas características, como los filtros de umbrales adaptativos.
- Se ha mejorado el rendimiento de la aplicación, incluyendo la programación multihilo y el cambio de algunos cálculos para que fueran más rápidos.
- Se ha mejorado la *GUI* (*Intefaz Gráfica de Usuario*), haciéndola más intuitiva y funcional.
- Se ha mejorado el diseño de la aplicación, buscando que sea más fácil de ampliar y mantener.
- Se ha mejorado la documentación del código.
- Se ha mejorado la calidad del código.
- Se han incluido nuevas funcionalidades, como el cálculo de características geométricas y la tabla de resultados, que permite interactuar con los defectos dibujados.
- Durante todo el proceso se han reforzado conceptos y técnicas tratadas durante la carrera.
- Se ha perfeccionado el conocimiento sobre *Java*, como por ejemplo, mediante la programación multihilo.
- Se ha aprendido un nuevo modo de realizar documentos técnicos con el uso de LaTeX. Aunque en un principio supuso una carga a la documentación, a medida que avanzó el desarrollo, favoreció el interés por la escritura debido a los retos que se presentaron durante su realización.
- Se ha podido aplicar y, así, obtener un conocimiento más profundo, una metodología de desarrollo ágil como es *Scrum*, tan en auge en la actualidad.
- Por último, destacar el perfeccionamiento de todas las tareas del desarrollo software: planificación, análisis, diseño, implementación, pruebas y documentación

Por todo ello, consideramos cumplidos los objetivos del proyecto.

6.3 Líneas de trabajo futuras

Este proyecto representa un prototipo mejorado de una herramienta que debe evolucionar y mejorar con los años. Además, el análisis de la bibliografía y de los conceptos teóricos supone un esfuerzo que puede facilitar el trabajo de los alumnos que continúen el desarrollo.

Durante la fase de diseño se ha tenido muy en cuenta la realización de una aplicación que permita posteriores ampliaciones y mejoras. Se ha buscado crear una base sobre la que se pueda seguir como referencia a la hora de realizar trabajos parecidos o ampliar el mismo. Ha habido algunas ideas que no han podido ser incorporadas debido a la falta de tiempo, y que podrían ser implementadas en un futuro para ampliar el proyecto. A continuación se proponen algunas:

- Clasificar los defectos en tipos: actualmente, la aplicación únicamente detecta los defectos, pero no los clasifica. Sería interesante conseguir que, una vez detectado el defecto, se informara al usuario del tipo al que pertenece. La inclusión del cálculo de características geométricas debería facilitar esto.
- Cálculo de nuevas características: Se podrían añadir más características, como los Filter Banks, para intentar mejorar la detección aún más.
- Inclusión de una nueva forma de detectar defectos, propuesta por nuestros tutores, José Francisco Díez Y César I. García, en la que primero se utiliza el filtro de umbrales locales ya visto en esta memoria para detectar regiones candidatas a albergar defecto, sobre las cuales se aplican después los cálculos de características, sin utilizar ventanas. Esta aproximación ha demostrado ser muy rápida, pero en ocasiones no se comporta bien. Por ello, se podría incluir en el proyecto y, de forma inteligente, determinar cuál es el mejor método a utilizar.
- Adaptación de la aplicación para que pueda ser ejecutada en un supercomputador.
- Otras aplicaciones: Se podría intentar utilizar las técnicas utilizadas en este proyecto para resolver otros problemas parecidos, como por ejemplo el descrito en el artículo Automated fish bone detection using X-ray imaging [36], también de Domingo Mery. En este trabajo se utiliza la misma metodología para detectar espinas de pescado.

Universidad de Burgos

Escuela Politécnica Superior

Ingeniería Informática

Área de Lenguajes y Sistemas Informáticos

Anexo I - Plan del proyecto software

X-RayDetector: Detección automática de defectos en piezas metálicas mediante análisis de radiografías

Adrián González Duarte y Joaquín Bravo Panadero

Directores: Dr. César I. García Osorio
José Francisco Díez Pastor

A. PLAN DEL PROYECTO SOFTWARE

A.1 Introducción

En este anexo se detalla el estudio desde el punto de vista temporal y de la viabilidad del proyecto software.

La planificación es una de las tareas más importantes en el desarrollo de un proyecto software y servirá para determinar objetivos, evaluar la viabilidad del proyecto, priorizar actividades...

En la primera parte del anexo se detallará la planificación temporal del proyecto teniendo en cuenta la metodología ágil que se va a utilizar: *Scrum*. En esta fase se determinarán los elementos que forman el *Product Backlog* y la prioridad de cada uno de ellos.

Debido a la metodología empleada, no se utilizará el clásico diagrama de *GANTT*. En lugar de esto, se definirá el *Product Backlog* y para el seguimiento se utilizará una herramienta de gestión especializada en metodologías ágiles, *PivotalTracker*, que permite el seguimiento diario de las tareas por parte del equipo de desarrollo.

En la segunda parte se calcularán los costes, analizando la rentabilidad del proyecto y justificando su desarrollo desde diversos puntos de vista: viabilidad técnica, legal, económica...

A.2 Planificación temporal del proyecto

Como se explicó en la memoria, para el desarrollo del proyecto de final de carrera se va a utilizar una metodología ágil llamada *Scrum*. Esta metodología establece una serie de prácticas que serán llevadas a cabo con algunas limitaciones debido al reducido tamaño del equipo de desarrollo.

Para poder estimar de manera general el tiempo total que va a llevar el desarrollo del proyecto se va a realizar una estimación a partir de los casos de uso definidos en el Anexo II.

A.2.1 Estimación temporal a partir de casos de uso

Antes de mostrar la tabla de estimación temporal, conviene repasar una serie de fórmulas que se utilizan para calcular algunos valores de la tabla:

- Puntos de casos de uso no ajustados:

$$UUCP = \text{PesoDeActores} + \text{PesoDeCasosDeUso}$$

- Peso de los casos de uso:

$$\text{PesoDeCasosDeUso} = \sum_{i=0}^{i} Factor_i$$

- Peso de los factores técnicos:

$$TFC = 0,6 + 0,01 \cdot \sum_{i=0}^{i} Factor_i \cdot Peso_i$$

- Factores de entorno:

$$EF = 1,4 + (-0,03) \cdot \sum_{i=0}^{i} Factor_i \cdot Peso_i$$

A continuación (ver tabla A.1), aparece detallada la duración estimada del desarrollo del proyecto a partir de los casos de uso identificados. Se encuentra dividida en:

- Puntos de casos de uso no ajustados: sirven para conocer la envergadura del proyecto tomando como referencia los casos de uso.
- Factores técnicos: cuantifica la dificultad del proyecto en función de sus características internas.
- Factores del entorno: sirven para valorar lo familiarizado que se encuentra el equipo de desarrollo con proyectos de este tipo.

Los puntos de casos de uso se calculan según la siguiente fórmula:

$$UCP = UUCP \cdot TF \cdot EF$$

Para calcular los puntos de casos de uso hay que sustituir, en la fórmula anterior, con los valores de la tabla, es decir:

$$UCP = 68 \cdot 0,89 \cdot 0,95 = 57,494$$

Factor	Peso	F_i	Total
Actores simples	0	1	0
Actores medios	0	2	0
Actores complejos	1	3	3
Total peso actores			3
Casos de uso simples	5	5	25
Casos de uso medios	1	10	10
Casos de uso complejos	2	15	30
Total peso casos de uso			65
UUCP (Puntos de caso de uso no ajustados)			68
Sistema distribuido	1	2	2
Tiempos de respuesta críticos	1	1	1
En línea	1	1	1
Procesos internos complejos	5	1	5
El código debe ser reutilizable	3	1	3
Fácil de instalar	1	0,5	0,5
Fácil de utilizar	5	0,5	2,5
Portable	1	2	2
Fácil de modificar	4	1	4
Concurrencia	1	1	1
Incluye características de seguridad	1	1	1
Acceso a software creado por otras compañías	1	1	1
Incluye facilidades de aprendizaje para usuario	5	1	5
TF (Factores técnicos)			0,89
Familiarizado con *Scrum*	1	0,5	0,5
Experiencia en este tipo de aplicaciones	2	1	2
Experiencia en Orientación a Objetos	5	0,5	2,5
Capacidad de liderazgo del analista	5	1	5
Motivación	5	1	5
Requisitos estables	3	2	6
Trabajadores a tiempo parcial	5	-1	-5
Lenguaje de programación difícil de utilizar	1	-1	-1
EF (Factores de entorno)			0,95

Tabla A.1.: Estimación temporal a partir de casos de uso

Para obtener la duración del proyecto estimado según los casos de uso hay que multiplicar el valor de *UCP* por un factor que depende del número de factores de entorno (*EF*) a los cuales se les haya dado peso 0. En este caso, como no se ha dado ningún valor cero se multiplica por 20.

Por lo expuesto anteriormente: $N^{o}horas = 20 \cdot 57,494$, es decir, 1150 horas/hombre. [CAMBIAR]

Conviene destacar que los resultados obtenidos son meramente orientativos, ya que se va a utilizar una metodología ágil, por lo que la planificación va a ser a nivel de *Sprint* (de 15 a 30 días). Al comenzar cada uno de ellos, se definirán una serie de tareas que deberán ser completadas en dicho *Sprint*.

El proyecto comenzará en noviembre de 2012 y se desea finalizar en junio de 2013, es decir, se trabajará durante 7 meses. Se deben realizar los cálculos de horas para comprobar si es viable el desarrollo en las fechas previstas:

$$\text{Tiempo} = \frac{1150\,\text{horas}}{1\,\text{persona}} \cdot \frac{1\,\text{jornada}}{7\,\text{horas}} \cdot \frac{1\,\text{mes}}{20\,\text{jornadas}} = 8,21\,\text{meses}$$

El análisis de casos de uso junto con el horario marcado hace que la fecha de finalización deba trasladarse hasta julio para poder cumplir los plazos. De este modo la fecha de finalización del proyecto se retrasa hasta julio de 2010, manteniendo la jornada laboral de 35 horas semanales.

[TODO ESTO HAY QUE CAMBIARLO, PERO NOS SIRVE DE PLANTILLA]

A.3 Aplicando una metodología ágil: *SCRUM*

En esta sección se repasa de una manera rápida los principales aspectos del desarrollo de proyectos con *Scrum*. Al mismo tiempo se analizan los distintos roles que va a asumir cada uno de los participantes en el proyecto de final de carrera.

Conviene aclarar que los términos que define la metodología *Scrum* se encuentran en inglés, han sido mantenidos en el idioma original para no confundir con las traducciones.

Scrum [11] es una metodología para la gestión y desarrollo de proyectos software basada en un proceso iterativo e incremental. Cada iteración termina con una pieza de software ejecutable que incorpora una nueva funcionalidad o mejora las ya existentes. Estas iteraciones suelen durar de dos a cuatro semanas.

Scrum busca priorizar los trabajos que mayor valor aportan al negocio evitando, en la medida de lo posible, complejos manuales de documentación que no tengan utilidad en el proceso.

Los requerimientos y prioridades se revisan y ajustan durante el proyecto en intervalos cortos y regulares. De esta forma es sencillo adaptarse a los cambios solicitados por el cliente y responder de una forma rápida a los mismos.

A.3.1 Actores

Scrum define un conjunto de roles o actores detallados a continuación:

- *Scrum Master* (o facilitador): coordina el desarrollo del proyecto y trabaja de manera similar al director de proyectos. Una de sus tareas es eliminar los obstáculos que puedan dificultar al equipo de desarrollo o *team* la consecución de sus objetivos. En el proyecto, este rol lo han tomado los autores del mismo: Adrián González y Joaquín Bravo.
- *Scrum Team* (o equipo): incluye a los desarrolladores y son los encargados de realizar las tareas que se definen en cada *Sprint*. En el proyecto, este rol lo han tomado los autores del mismo.
- *Product Owner* (o cliente): representa la voz del cliente y aporta la visión del negocio. Representa el destinatario final del proyecto a desarrollar y el que, en última instancia, realiza las pruebas de aceptación. Es también el encargado de mantener al día el listado de las tareas o *Product Backlog* y sus prioridades. En el proyecto este rol sería tomado por la Universidad de Burgos, en concreto, personalizado en los tutores del proyecto César I. García Osorio y José Francisco Díez Pastor.

A.3.2 Ciclo de desarrollo

Al inicio del proyecto se definen una serie de requisitos que serán los objetivos a cumplir. Todos ellos quedan reflejados en el *Product Backlog*. Cada uno de estos objetivos serán subdivididos en tareas pequeñas y atómicas al inicio de cada *Sprint*.

El ciclo de desarrollo, como se muestra (ver figura A.1), es iterativo a nivel de *Sprint*, al inicio del mismo se extraen una serie de tareas de los elementos del *Product Backlog* que conforman el *Sprint Backlog*. Estas tareas deberán ser completadas durante el ciclo.

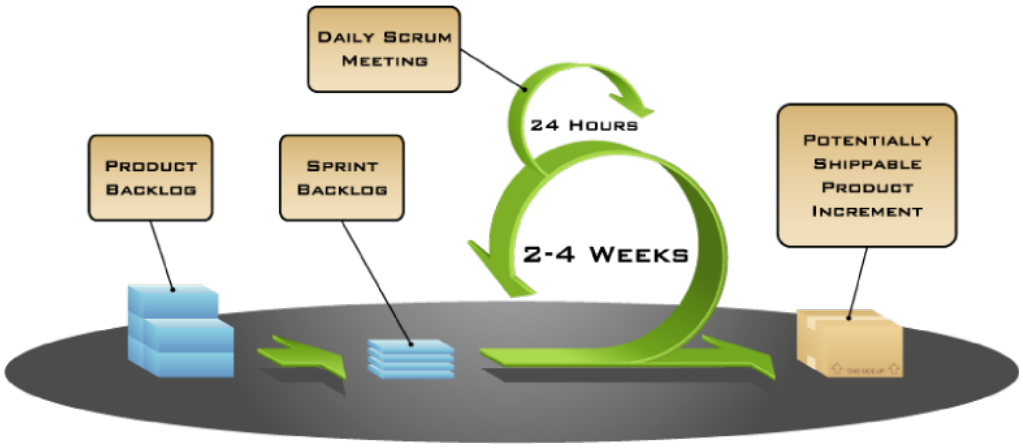

Figura A.1.: Metodología Scrum

Scrum define que debe realizarse una breve reunión diaria, *Daily Meeting*, en la que cada miembro del equipo de desarrollo explica lo que ha hecho desde la última sincronización, que va a hacer a partir de ese momento y las dificultades encontradas o que espera encontrar. En este caso, debido a problemas de tiempo y problemas para quedar, no se harán explícitamente estas reuniones.

Los *Sprints* se definirán, como establece *Scrum*, con una duración de dos a cuatro semanas que podrán alargarse o contraerse para coincidir con los tutores para las reuniones periódicas que establece:

- *Planning meeting*: reunión inicial de cada *Sprint* en la que se extraen los *item backlog*. Se numeran las tareas extraídas del *Product Backlog* para desarrollar durante el *Sprint*. Esta reunión tiene una gran importancia ya que, durante el desarrollo del *Sprint*, no se podrá modificar ni añadir nuevas tareas.
- *Review meeting*: reunión final (4 horas máximo) de cada *Sprint* donde se detallan los objetivos cumplidos durante el mismo. Se muestra al usuario el producto, en caso de que sea posible. En función de los resultados mostrados y de los cambios que haya habido en el contexto del proyecto, el cliente realiza las adaptaciones necesarias de manera objetiva, ya desde la primera iteración, replanificando el proyecto.
- *Retrospective meeting*: en esta reunión (4 horas máximo), el equipo analiza cómo ha sido su manera de trabajar y cuáles son los problemas que podrían impedirle progresar adecuadamente, mejorando de manera continua su productividad. El *Scrum Master*, o *Facilitador*, se encargará de ir eliminando los obstáculos identificados.

En este caso, todas las reuniones requieren de la presencia de los tutores y del equipo de desarrollo ya que entre ambos, agrupan los roles que se ven implicados en dichas reuniones.

A.4 Product Backlog del proyecto

En esta sección, aparece redactado el *Product Backlog* (ver tabla A.2) o lista de objetivos, con la prioridad de cada uno de los *Product Item*, el *sprint* al que pertenecen y la estimación que les dimos en su momento, basándonos en una escala de puntos tipo Fibonacci (1, 2, 3, 5, 8). Hemos decidido incluir también los *bugs* que se han ido localizando[INSERTAR CITA].

Scrum no define un método o herramienta para llevar el día a día del trabajo realizado, por lo que el seguimiento se puede realizar con una simple hoja de cálculo, o con herramientas de gestión especializadas en este tipo de desarrollos.

En este caso, se ha utilizado *PivotalTracker* (https://www.pivotaltracker.com/) para el seguimiento. Es un software potente al que se accede mediante un navegador y sirve de control de tareas y defectos. Su licencia ha sido gratuita por tratarse de un proyecto público.

ID	Backlog Item	Propietario	Prioridad	Estimación	Sprint
S-01001	Se debe poder cargar una imagen y sacar información de sus píxeles	César, José	Alta	2	Sprint 1
S-01002	Se debe poder analizar una imagen y extraer sus características	César, José	Alta	5	Sprint 2
S-01003	La aplicación debe ser capaz de calcular una imagen Saliency Map	César, José	Alta	5	Sprint 3
S-01004	La aplicación debe poder calcular las características de Haralick	César, José	Alta	8	Sprint 3
S-01005	La aplicación debe ser capaz de generar ficheros ARFF	César, José	Alta	5	Sprint 3
S-01006	La aplicación debe ser capaz de calcular los eigenvalues mediante weka.Matrix o COLT	César, José	Baja	1	Sprint 4
S-01007	La aplicación debe ser capaz de dividir la imagen según los hilos disponibles, con solapamiento entre las mismas	César, José	Alta	1	Sprint 4
S-01008	La aplicación debe ser capaz de realizar la convolución de la imagen completa, para extraer la media	César, José	Media	3	Sprint 4

continúa en la página siguiente

continúa desde la página anterior

ID	Backlog Item	Propietario	Prioridad	Estimación	Sprint
S-01009	La aplicación debe ser capaz de albergar varios tipos de ventana	César, José	Alta	1	Sprint 4
S-01010	La aplicación debe ser capaz de entrenar un clasificador y clasificar	César, José	Alta	5	Sprint 4
S-01011	La aplicación debe ser capaz de mostrar un GUI con las principales opciones	César, José	Alta	5	Sprint 4
S-01012	La aplicación debe mostrar el progreso en una Progress Bar	César, José	Baja	2	Sprint 5
S-01013	La aplicación debe mostrar un aviso si no se selecciona una región de la imagen a analizar	César, José	Baja	1	Sprint 5
S-01014	La ventana de resultados se debe limpiar después de cada análisis	César, José	Media	1	Sprint 5
S-01015	La aplicación debe escribir un ARFF por cada hilo, uniéndolos al terminar el proceso	César, José	Alta	3	Sprint 5
S-01016	La aplicación debe mostrar eventos de funcionamiento en la interfaz	César, José	Media	2	Sprint 5
S-01017	La aplicación debe guardar y cargar en un fichero las opciones	César, José	Media	3	Sprint 5
S-01018	La aplicación debe ser capaz de guardar un log de errores	César, José	Baja	2	Sprint 5
S-01019	La aplicación debe usar el filtro saliency para calcular características para entrenar y clasificar	César, José	Alta	3	Sprint 5
S-01020	La aplicación debe usar la convolución de la imagen completa, en el caso de que no se elija región, o toda la región más un margen	César, José	Alta	3	Sprint 5
S-01021	La aplicación debe poder dibujar los defectos encontrados	César, José	Alta	3	Sprint 5
S-01022	La aplicación debe ser capaz de usar varios tipos de métodos para determinar si una ventana es defectuosa o no	César, José	Alta	8	Sprint 5

continúa en la página siguiente

continúa desde la página anterior

ID	Backlog Item	Propietario	Prioridad	Estimación	Sprint
S-01023	La aplicación debe mostrar un slider para seleccionar el umbral de detección	César, José	Alta	3	Sprint 5
S-01024	La aplicación debe ser capaz de escribir texto formateado en HTML en el panel de log	César, José	Baja	3	Sprint 6
S-01025	La aplicación debe ser capaz de exportar el log en formato HTML	César, José	Baja	2	Sprint 6
S-01026	La aplicación debe tener un menú con varias opciones	César, José	Media	2	Sprint 6
S-01027	La aplicación debe usar REPTree para regresión lineal	César, José	Media	1	Sprint 6
S-01028	La aplicación debe ser capaz de calcular filtros de umbrales locales y comprobar si las regiones candidatas están dentro de la máscara	César, José	Alta	5	Sprint 6
S-01029	La aplicación debe dirigir la búsqueda de defectos usando filtros de umbrales locales	César, José	Alta	5	Sprint 6
B-01001	BUG: la primera y segunda derivadas deben estar bien calculadas	César, José	Alta	-	Sprint 6
S-01030	La aplicación debe dar la oportunidad al usuario de elegir el proceso de detección de defectos	César, José	Media	2	Sprint 7
S-01031	La aplicación debe calcular los umbrales locales con un pequeño margen	César, José	Media	2	Sprint 7
S-01032	La aplicación debe ser capaz de generar un conjunto de datos a partir de la ventana deslizante	César, José	Alta	3	Sprint 7
S-01033	La aplicación debe poder remuestrear la lista de píxeles blancos, de tal manera que no considere todos	César, José	Alta	5	Sprint 7

continúa en la página siguiente

continúa desde la página anterior

ID	Backlog Item	Propietario	Prioridad	Estimación	Sprint
S-01034	La aplicación debe ser capaz de calcular características geométricas sobre las regiones segmentadas	César, José	Alta	5	Sprint 7
S-01035	Se debe poder mostrar una tabla resumen de los defectos	César, José	Alta	5	Sprint 7
S-01036	La aplicación debe permitir al usuario elegir qué tipo de clasificación se va a usar: clases nominales o regresión	César, José	Media	2	Sprint 8
S-01037	La aplicación debe añadir un borde a la selección en aquellas zonas que no coincidan con los bordes de la imagen	César, José	Media	3	Sprint 8
S-01038	La aplicación debe permitir al usuario elegir qué tipo de ventana defectuosa quiere usar	César, José	Media	3	Sprint 8
S-01039	Se debe añadir un módulo de ayuda en línea (JavaHelp), accesible desde el menú y con teclas rápidas (F1)	César, José	Media	3	Sprint 8
S-01040	La aplicación debe poder permitir seleccionar una fila de la tabla de resultados e iluminar el defecto correspondiente	César, José	Alta	5	Sprint 8
S-01041	La aplicación debe dar la opción de elegir si se calculan todas las características o sólo las mejores	César, José	Media	3	Sprint 9
S-01042	La aplicación debe tratar las excepciones de forma correcta, mostrando avisos	César, José	Media	3	Sprint 8
B-01002	BUG: la aplicación debe impedir la reselección mientras se analiza	César, José	Alta	-	Sprint 8
S-01043	La aplicación debe permitir guardar la imagen final	César, José	Media	2	Sprint 9
S-01044	La aplicación debe permitir guardar los defectos binarizados	César, José	Media	2	Sprint 9
S-01045	La aplicación debe calcular «precision and recall» del resultado	César, José	Media	3	Sprint 9

continúa en la página siguiente

continúa desde la página anterior

ID	Backlog Item	Propietario	Prioridad	Estimación	Sprint
B-01003	BUG: coordenadas en lista de píxeles blancos	César, José	Alta	-	Sprint 9

Tabla A.2.: Product Backlog

A.5 Planificación por Sprint

En esta sección aparece el listado de los diversos *Sprints* del proyecto. La información de cada uno está dividida del siguiente modo:

- *Planning meeting*: acta de la reunión inicial de cada *Sprint* en la que se detallan los objetivos a cumplir durante el transcurso del mismo.
- *Sprint planning*: se numeran las tareas extraídas del *Product Backlog* para desarrollar durante el *Sprint*.
- *Burndown chart*: gráfico que muestra la evolución a lo largo del *Sprint* en comparación a la línea óptima.
- *Retrospective meeting*: acta de la reunión retrospectiva realizada al final del *Sprint* y donde se analizan los problemas detectados durante su ejecución y las desviaciones (en caso de haberlas).

Debido a la falta de conocimiento inicial y a la incertidumbre sobre el desarrollo, se inicia el proyecto con *Sprints* de corta duración, aproximadamente dos semanas. Se ha intentado mantener siempre esta duración, aunque a veces nos ha podido el optimismo y hemos metido demasiado trabajo o no se han tenido en cuenta ciertos problemas de tiempo.

Las reuniones inicial y final de cada *Sprint* se solaparon para poder hacerlas el mismo día, debido a problemas de tiempo y de disponibilidad. Así, media parte de esta reunión se centraba en la *Planning meeting* y la otra en la *Retrospective meeting*.

A.5.1 Sprint 1: Spike Arquitectónico 1

■ Planning meeting

En este primer *Sprint* se pretende comenzar el proyecto, empezar a usar ImageJ, familiarizarnos con el proyecto del año pasado... Se empiezan a poner en práctica algunas de las ideas que teníamos para el diseño de la aplicación. Lo llamamos *spike arquitectónico* debido a que es una primera aproximación al proyecto, tomando prestado el nombre de otra metodología ágil (*Extreme Programming*).

■ Fechas de desarrollo y tiempo estimado

Este primer *Sprint* comienza el 19/11/2012 y su fecha límite es el 27/11/2012. Es un *Sprint* muy corto debido a que es una primera aproximación. Los puntos totales estimados en este *Sprint* son de 2.

■ Sprint planning

- S-01001: Se debe poder cargar una imagen y sacar información de sus píxeles.
  - TK-00001: Carga y manejo de la imagen con ImageJ.
  - TK-00002: Creación de la ventana deslizante (simple).

- TK-00003: Creación de una pequeña interfaz que muestre los resultados.

■ Burndown chart

Para este *Sprint* no disponemos de *Burndown Chart* debido a un fallo en el uso de *Pivotal Tracker*. Como aún no estábamos familiarizados con él, finalizamos antes la *release* que la historia de usuario, con lo que no se ve reflejado en el *Burndown Chart*.

■ Retrospective meeting

¿Qué ha ido bien?

- La lectura de artículos y comprensión del proyecto avanza por buen camino.
- Nos vamos familiarizando con ImageJ y el resto de nuevas herramientas.

¿Qué dificultades ha habido?

- Familiarizarse con ImageJ es difícil. Por ello, llegamos a plantearnos el uso de alguna otra librería, como *OpenCV*, pero acabamos descartándolo debido a que ImageJ tienen muy buena documentación, cosa que el resto no.
- Adaptación a la nueva metodología.

A.5.2 Sprint 2: Spike Arquitectónico 2

■ Planning meeting

Como seguimos probando cosas más que implementar funcionalidad nueva, consideramos a este *Sprint* otro *spike*, aunque éste está ya más orientado a nueva funcionalidad. Básicamente, lo que buscamos en este *Sprint* es:

1. Seguir familiarizándonos con ImageJ.
2. Convertir la aplicación en multihilo, dividiendo la imagen a analizar en tantas partes como procesadores tenga la máquina y analizando cada trozo en un hilo separado.
3. Implementar los algoritmos de extracción de características, usando lo que se pueda del proyecto del año pasado.

También comentamos otras cosas como que, de momento, vamos a sacar los resultados por pantalla, pero que hay que ir pensando en sacarlos a un fichero. También tenemos que pensar que hay que usar *Saliency Map* para el preprocesamiento.

■ Fechas de desarrollo y tiempo estimado

Este *Sprint* comienza el 27/11/2012 y la fecha límite es el 11/12/2012. El número total de puntos estimado es de 5.

Figura A.2.: Burndown Chart del segundo Sprint

Sprint planning

- S-01002: Se debe poder analizar una imagen y extraer sus características.
  - TK-00004: Implementación de la ventana deslizante multihilo.
  - TK-00005: Implementación de los algoritmos de selección de características.
  - TK-00006: Implementación de las llamadas a los algoritmos de selección de características y muestra de los resultados.

Burndown chart

En el *Burndown Chart* (ver figura A.2) podemos ver 2 cosas:

1. Hay una incoherencia en las fecha de inicio, ya que, de nuevo, por la falta de conocimientos del uso de *PivotalTracker*, metimos una historia de usuario antes de tiempo y después nos limitamos a cambiar el nombre, dándonos cuenta mucho después de que lo habíamos hecho mal.
2. Tuvimos ciertos problemas de tiempo y de implementación, por lo que hay un tiempo totalmente plano en el que no pudimos terminar ninguna tarea, lo que llevó a pasarnos de fecha hasta el 18/12/2012.

Retrospective meeting

¿Qué ha ido bien?

- Cada vez dominamos más ImageJ.
- Hemos conseguido implementar con éxito el multihilo.
- Las implementaciones de los algoritmos de extracción de características del año pasado parece que van bien, por lo que decidimos mantenerlas.
- El diseño que habíamos pensado está probando ser bueno.

¿Qué dificultades ha habido?

- Hemos planificado mal este *Sprint*, ya que fuimos muy optimistas respecto al tiempo y a las dificultades. Tuvimos mucha carga de trabajo de otras asignaturas.
- Tuvimos problemas con las coordenadas, lo que nos quitó mucho tiempo.

A.5.3 Sprint 3

■ Planning meeting

En este *Sprint* lo que se pretende es:

1. Implementar las características de Haralick, que por ser muy pesadas, no se hicieron en el anterior *Sprint*.
2. Calcular el *Saliency Map* para el preprocesamiento de la imagen.
3. Generar un fichero ARFF con el resultado de los análisis.

Como vemos, algunas de las cosas que se hablaron en anteriores reuniones se ponen en práctica en este *Sprint*.

■ Fechas de desarrollo y tiempo estimado

Este *Sprint* se planificó de manera especial, ya que por medio están las vacaciones de Navidad y el periodo de exámenes del primer cuatrimestre. Por eso es más largo de lo normal: del 19/12/2012 al 24/01/2013. El número total de puntos estimado es de 18.

■ Sprint planning

- S-01003: La aplicación debe ser capaz de calcular una imagen Saliency Map.
  - TK-00007: Implementar el cálculo de Saliency Map.
  - TK-00008: Adición de los Saliency Map a la estructura del programa.
- S-01004: La aplicación debe poder calcular las características de Haralick.
  - TK-00009: Implementar características de Haralick.
  - TK-00010: Añadirlo a la estructura del patrón Estrategia de algoritmos.
- S-01005: La aplicación debe ser capaz de generar ficheros ARFF.

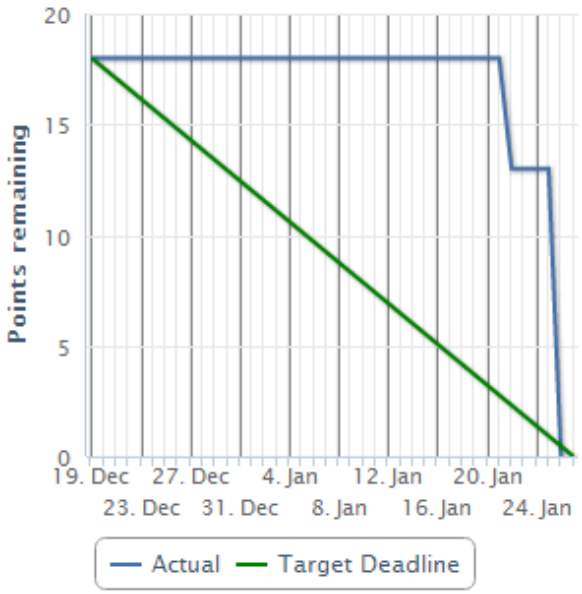

Figura A.3.: Burndown Chart del tercer Sprint

- TK-00011: Generación de ARFF a partir de los datos de las características.

■ Burndown chart

En el *Burndown Chart* (ver figura A.3) podemos ver que hay un gran periodo totalmente plano, en el que no pudimos prácticamente hacer nada del proyecto. Esto estaba previsto y controlado, y por eso planificamos este *Sprint* con tanto tiempo.

Por lo demás, vemos que en este *Sprint* no nos pasamos de la fecha límte.

■ Retrospective meeting

¿Qué ha ido bien?

- Los algoritmos de la versión anterior parecen funcionar bien, por lo que decidimos mantenerlo (excepto una cosa, de la que hablaremos más tarde).
- El diseño que habíamos pensado vuelve a probar ser bueno, ya que permite añadir nuevos algoritmos muy fácilmente.

¿Qué dificultades ha habido?

- La implementación de la última característica de Haralick es muy lenta. Hemos observado que los alumnos del año pasado hicieron su propia clase de cálculos de matrices, cosa que nos pareció absurda y contraproducente. De hecho, ellos anularon esta característica para presentar el proyecto.

- En este *Sprint* quizás nos pasamos estimando el tiempo de cada tarea, debido en parte a lo que nos pasó en el anterior *Sprint*.

A.5.4 Sprint 4

■ Planning meeting

En este *Sprint* planeamos hacer:

- Probar implementaciones alternativas para los autovalores de una matriz (Weka, COLT...).
- Añadir un solapamiento al dividir las imágenes para el multihilo, para tener en cuenta la importancia de los píxeles de la división, ya que pueden ser importantes en la otra parte de la imagen.
- Modificar la convolución de las imágenes para hacerlo con la imagen completa.
- Entrenar un clasificador, para lo que hace falta una ventana aleatoria.
- Implementar la interfaz de usuario.

Además, se habló de otras cosas, como el ir probando los umbrales adaptitivos en ImageJ, o cosas que deben ir en la memoria (la metodología debe aparecer en el apartado de técnicas, resaltar los aspectos relevantes y nuevos del proyecto de este año *versus* la versión del año pasado...). Llegamos incluso a hablar de la idea de clasificar los defectos en tipos, para lo cual es necesario calcular características geométricas.

■ Fechas de desarrollo y tiempo estimado

Este *Sprint* se inicia el 26/01/2013 y tiene como fecha límite el 15/02/2013. Tiene un total de puntos estimados de 20.

■ Sprint planning

- S-01006: La aplicación debe ser capaz de calcular los valores propios mediante weka.Matrix o COLT.
  - TK-00012: Añadir las librerías necesarias.
  - TK-00013: Sustituir la llamada a la función de cálculo de valores propios por el método correspondiente de las nuevas librerías.
- S-01007: La aplicación debe ser capaz de dividir la imagen según los hilos disponibles, con solapamiento entre las mismas.
  - TK-00014: Modificar la división de las imágenes para permitir un solapamiento.
- S-01008: La aplicación debe ser capaz de realizar la convolución de la imagen completa, para extraer la media.
  - TK-00015: Calcular la convolución de la imagen completa.

Figura A.4.: Burndown Chart del cuarto Sprint

- TK-00016: Calcular sobre la convolución la primera y segunda derivadas.

- S-01009: La aplicación debe ser capaz de albergar varios tipos de ventana.
  - TK-00017: Crear el patrón estrategia.
  - TK-00018: Creación de la clase de la ventana aleatoria.
- S-01010: La aplicación debe ser capaz de entrenar un clasificador y clasificar.
  - TK-00019: Crear 2 listas de píxeles centrales: una con defectos y otra sin defectos.
  - TK-00020: Seleccionar de forma aleatoria ventanas sobre las que calcular características, a partir de las 2 listas.
  - TK-00021: Entrenar a un clasificador y guardar el modelo obtenido.
  - TK-00022: Clasificar defectos.
- S-01011: La aplicación debe ser capaz de mostrar un GUI con las principales opciones.
  - TK-00023: Implementación de la GUI.
  - YK-00024: Adaptación del resto del código.

Burndown chart

En el *Burndown Chart* (ver figura A.4) podemos ver que el desarrollo fue bastante escalonado, teniendo periodos planos en los que no podíamos avanzar debido a las dificultades.

Todo esto al final hizo que nos pasáramos de fecha, ya que acabamos el 21 de febrero.

■ Retrospective meeting

¿Qué ha ido bien?

- El diseño que habíamos pensado vuelve a probar ser bueno, ya que, usando la misma idea para las ventanas que para los algoritmos, nos permite añadir nuevos tipos de ventana fácilmente.
- El entrenamiento funciona bien. Esto nos permitió ganar mucha comprensión acerca de cómo funciona esta parte del proyecto.
- La GUI implementada es más intuitiva que la del año pasado.

¿Qué dificultades ha habido?

- El cálculo de los autovalores siguen siendo muy lentos.
- Mala planificación: nos pasamos metiendo historias de usuario en este *Sprint*, por lo que nos ha llevado algo más de tiempo del previsto.
- Fallo extraño al crear el ARFF de entrenamiento: hay veces que los hilos no lo hacen bien.

A.5.5 Sprint 5

■ Planning meeting

En este *Sprint* planeamos hacer:

- Mejorar ciertos aspectos de la interfaz (barra de progreso, eventos en el log...).
- Solucionar el fallo al escribir los ARFF (decidimos que cada hilo escriba en un fichero y luego se fusionen).
- Dibujar los defectos encontrados.
- Solucionar ciertos aspectos que no habían quedado claros respecto a los *Saliency Map* y a las convoluciones.
- Implementar nuevas formas de determinar cuándo una ventana es defectuosa o no en el entrenamiento.
- Implementar la selección del umbral de detección de defectos.
- Implementar la carga y almacenamiento de opciones.

Las nuevas formas de determinar cuándo una ventana es defectuosa son las que ya mencionamos en la parte teórica de esta memoria:

- Un píxel mal, toda la ventana mala (enfoque del año pasado).
- Píxel central malo, toda la ventana mala.
- Porcentaje de píxeles malos.
- Porcentaje de píxeles malos en una región de vecinos 3×3 centrada en el píxel central.

Además, se habla de analizar si es mejor *Random Forest*(que es un método de construcción de ensembles[REFERENCIA]) que *Bagging* y la posibilidad de incluir un árbol para poder seleccionar el algoritmo de clasificación.

■ Fechas de desarrollo y tiempo estimado

Este *Sprint* se inicia el 21/02/2013 y tiene como fecha límite el 12/03/2013. Tiene un total de puntos estimados de 34.

■ Sprint planning

- S-01012: La aplicación debe mostrar el progreso en una Progress Bar.
  - TK-00025: Actualizar el progreso de la Progress Bar en el análisis de la imagen.
  - TK-00026: Actualizar el progreso de la Progress Bar en el entrenamiento.
- S-01013: La aplicación debe mostrar un aviso si no se selecciona una región de la imagen a analizar.
  - TK-00027: Creación del aviso con 2 botones: Continuar y Cancelar el proceso.
- S-01014: La ventana de resultados se debe limpiar después de cada análisis.
  - TK-00028: Limpiar la ventana si se para la ejecución o si se vuelve a lanzar el proceso.
- S-01015: La aplicación debe escribir un ARFF por cada hilo, uniéndolos al terminar el proceso.
  - TK-00029: Generación de un ARFF nuevo en cada hilo, sobre el que va a escribir cada uno.
  - TK-00030: Fusión de todos los ARFF generados al terminar el proceso.
  - TK-00031: Entrenamiento del clasificador con el ARFF fusionado.
  - TK-00032: Mover la funcionalidad de generar el modelo a otra clase.
- S-01016: La aplicación debe mostrar eventos de funcionamiento en la interfaz.
  - TK-00033: Escritura en el campo de texto de la interfaz cada vez que ocurre un evento.
- S-01017: La aplicación debe guardar y cargar en un fichero las opciones.
  - TK-00034: Creación de un fichero de propiedades, con las opciones por defecto.
  - TK-00035: Creación de una clase de utilería que sea capaz de leer y escribir el fichero de opciones.
  - TK-00036: Sustituir algunos de los parámetros dentro del programa por llamadas a la clase de lectura/escritura de propiedades.
- S-01018: La aplicación debe ser capaz de guardar un log de errores.
  - TK-00037: Creación de una clase que escriba el log.

    - TK-00038: Escritura en el log en cada excepción.
- S-01019: La aplicación debe usar el filtro saliency para calcular características para entrenar y clasificar.
    - TK-00039: Uso de las imágenes saliency al mismo tiempo que las normales a la hora de calcular las características.
    - TK-00040: Generación de un ARFF con las características de las imágenes saliency.
- S-01020: La aplicación debe usar la convolución de la imagen completa, en el caso de que no se elija región, o toda la región más un margen.
    - TK-00041: Calcular el margen de la región y crear con ello una nueva imagen, para pasarla a la ventana.
- S-01021: La aplicación debe poder dibujar los defectos encontrados.
    - TK-00042: Creación de matriz de píxeles, en la que cada pixel será considerado como error si se supera el umbral.
    - TK-00043: Dibujado de los píxeles con defecto en una imagen nueva (máscara).
    - TK-00044: Superposición de la máscara con la imagen original para crear la nueva imagen.
    - TK-00045: Dibujado de la imagen en el panel de visualización.
- S-01022: La aplicación debe ser capaz de usar varios tipos de métodos para determinar si una ventana es defectuosa o no.
    - TK-00046: Implementación de «píxel central ->todo mal».
    - TK-00047: Creación del ARFF para que se pueda usar en regresión (valor de la clase numérico).
    - TK-00048: Implementación de «píxel central más tanto por ciento de vecinos».
    - TK-00049: Implementación de «porcentaje de todos los píxeles de la ventana».
    - TK-00050: Entrenamiento de un modelo de regresión lineal y usarlo para clasificar.
- S-01023: La aplicación debe mostrar un slider para seleccionar el umbral de detección.
    - TK-00051: Creación del slider, junto con su listener.

Burndown chart

En el *Burndown Chart* (ver figura A.5) podemos ver que el desarrollo fue bastante cercano a la recta ideal.

Esto hizo que, pese a tener mucha carga de trabajo en este *Sprint*, terminásemos dentro del tiempo. Fue una buena planificación.

Figura A.5.: Burndown Chart del quinto Sprint

■ Retrospective meeting

¿Qué ha ido bien?

- El dibujado de defectos del año pasado funciona muy bien, por lo que hemos decidido mantenerlo.
- La regresión lineal funciona.
- Es muy sencillo generar el fichero de opciones. Decidimos meter alguna más.
- Dos de las nuevas formas de determinar si una ventana es defectuosa han resultado ser muy buenas, por lo que se mejora la precisión respecto al año pasado.
- El log con colores queda más intuitivo que el del año pasado.
- Se solucionó el problema de la escritura en los ARFF.
- Buena planificación del *Sprint*.

¿Qué dificultades ha habido?

- No hemos encontrado ninguna forma de automatizar el valor del umbral de detección de defectos. Se deja, por lo tanto, con el slider sin más.
- La barra de progreso nos ha dado algunos problemas.

A.5.6 Sprint 6

■ Planning meeting

En este *Sprint* lo que se pretende es:

- Cambiar la escritura del log por HTML.
- Exportación del log.
- Crear un menú de opciones avanzadas, aprovechando las posibilidades que nos da el fichero de opciones.
- Arreglar un fallo con la primera y segunda derivadas.
- Implementar dos nuevas formas de enfocar la detección de defectos

Las nuevas formas de detección de defectos son las que ya se explicaron en la parte teórica de la memoria:

- Detección normal más intersección con filtros de umbrales locales.
- Lista de píxeles blancos en umbrales locales.

Además, también se habla de otras cosas, como que la documentación *Javadoc* debería ir en inglés y que se deben añadir todas las mejoras que hemos hecho a la presente memoria.

■ Fechas de desarrollo y tiempo estimado

Este *Sprint* se inicia el 12/03/2013 y tiene como fecha límite el 26/03/2013. Tiene un total de puntos estimados de 18.

■ Sprint planning

- S-01024: La aplicación debe ser capaz de escribir texto formateado en HTML en el panel de log.
  - TK-00052: Crear un CSS con una clase por cada tipo de cadena.
  - TK-00053: Sustituir las escrituras con SimpleAttributeSet por HTML.
- S-01025: La aplicación debe ser capaz de exportar el log en formato HTML.
  - TK-00054: Creación del botón junto con su listener.
  - TK-00055: Escritura del texto formateado en HTML en el fichero seleccionado.
- S-01026: La aplicación debe tener un menú con varias opciones.
  - TK-00056: Creación de la barra de menús, con sus diferentes apartados.
  - TK-00057: Creación del diálogo de «opciones avanzadas».
- S-01027: La aplicación debe usar REPTree para regresión lineal.

    - TK-00058: Sustituir LinearRegression por REPTree.
- S-01028: La aplicación debe ser capaz de calcular filtros de umbrales locales y comprobar si las regiones candidatas están dentro de la máscara.
    - TK-00059: Calcular filtros de umbrales locales.
    - TK-00060: Comprobar píxeles de regiones candidatas.
    - TK-00061: Segmentar el defecto.
- S-01029: La aplicación debe dirigir la búsqueda de defectos usando filtros de umbrales locales.
    - TK-00062: Calcular filtros de umbrales locales.
    - TK-00063: Sacar lista de píxeles marcados como blancos en los umbrales locales.
    - TK-00064: Dividir la lista en tantas partes como hilos haya.
    - TK-00065: Centrar una ventana en cada píxel de las regiones candidatas, calcular características y clasificar.
    - TK-00066: Segmentar el defecto con el algoritmo actual.
- B-01001: BUG: la primera y segunda derivadas deben estar bien calculadas.
    - TK-00067: Verificar, comprobar y arreglar la primera y segunda derivadas.

Burndown chart

En el *Burndown Chart* (ver figura A.6) podemos ver que el desarrollo fue cercano a la recta ideal.

Esto hizo que terminásemos dentro de tiempo. Es más, podríamos incluso haber metido más carga de trabajo, ya que terminamos algo antes de tiempo. La planificación no fue mala.

Retrospective meeting

¿Qué ha ido bien?

- El log en HTML queda muy bien y es muy sencillo de hacer.
- La exportación del log ha quedado mucho más fina que la del año pasado.
- Ha sido sencillo adaptar las clases de ImageJ para poder usar los umbrales locales.
- El uso de los umbrales locales nos da más conocimientos sobre análisis de imágenes y nos abre una nueva forma de poder solucionar el problema.
- Se solucionó el problema de la barra de progreso.
- Se refinó el cálculo de la primera y segunda derivadas usando métodos de ImageJ.
- El menú ha sido muy fácil de hacer y nos permite muchas opciones.
- Buena planificación del *Sprint*.

Figura A.6.: Burndown Chart del sexto Sprint

¿Qué dificultades ha habido?

- La segunda aproximación de la detección de defectos (lista de píxeles blancos) no devuelve tan buenos resultados como pensábamos.

A.5.7 Sprint 7

Planning meeting

En este *Sprint* lo que se pretende es:

- Permitir al usuario elegir más opciones.
- Mejorar el cálculo de los umbrales locales.
- Calcular características geométricas sobre los defectos dibujados.
- Mostrar una tabla resumen.
- Intentar encontrar una buena solución para el problema de los autovalores.
- Solucionar el problema de la lista de píxeles blancos.

Como vemos, se van a poner en práctica ideas ya comentadas en reuniones anteriores.

Fechas de desarrollo y tiempo estimado

Este *Sprint* se inicia el 26/03/2013 y tiene como fecha límite el 18/04/2013. Lo hemos planificado con una semana extra debido a que están las vacaciones de Semana Santa de por medio y nos puede dar problemas. Tiene un total de puntos estimados de 22.

Sprint planning

- S-01030: La aplicación debe dar la oportunidad al usuario de elegir el proceso de detección de defectos.
  - TK-00068: Adición de las opciones a la interfaz.
  - TK-00069: Lanzamiento de una opción u otra según la selección.
- S-01031: La aplicación debe calcular los umbrales locales con un pequeño margen.
  - TK-00070: Generación de la imagen a calcular, partiendo de la selección más el radio del algoritmo.
- S-01032: La aplicación debe ser capaz de generar un conjunto de datos a partir de la ventana deslizante.
  - TK-00071: Generación de un ARFF a partir de la ventana deslizante.
  - TK-00072: Inclusión de la opción en la interfaz.
- S-01033: La aplicación debe poder remuestrear la lista de píxeles blancos, de tal manera que no considere todos.
  - TK-00073: Sacar una lista de ROIs marcados como blancos en los umbrales locales.
  - TK-00074: Crear una regla para decidir si el píxel que toca realmente se va a considerar o no.
- S-01034: La aplicación debe ser capaz de calcular características geométricas sobre las regiones segmentadas.
  - TK-00075: Cambiar la llamada a ParticleAnalyzer para que calcule características geométricas.
  - TK-00076: Creación de un método para mostrar los resultados por cada ROI.
- S-01035: Se debe poder mostrar una tabla resumen de los defectos.
  - TK-00077: Generación de una JTable en la interfaz que contenga una fila por cada región.

Burndown chart

En el *Burndown Chart* (ver figura A.7) podemos ver que hubo una primera parte en la que la curva es totalmente plana. Coincide con las vacaciones.

Pese a ello, terminamos a tiempo. Esto es debido a que tuvimos en cuenta esta eventualidad, dando lugar a una buena planificación.

Figura A.7.: Burndown Chart del séptimo Sprint

■ Retrospective meeting

¿Qué ha ido bien?

- Se vuelve a ver que el fichero de opciones nos da muchas posibilidades para personalizar el funcionamiento de la aplicación.
- El cálculo de las características geométricas funciona perfectamente.
- Se ha encontrado una buena solución al problema de los autovalores con *EJML* (*Efficient Java Matrix Library*), aunque sigue siendo algo lento. Tendremos que decidir si lo usamos finalmente o no.

¿Qué dificultades ha habido?

- No se ha solucionado del todo el problema del remuestreo de píxeles blancos. Lo que hemos hecho es considerar aleatoriamente píxeles de cada región, hasta un máximo del 10 % del área de la misma. Esto provoca que se consideren muy pocos píxeles, por lo que habrá que probar con un porcentaje mayor.

A.5.8 Sprint 8

■ Planning meeting

En este *Sprint* lo que se pretende es:

- Dejar el código limpio y bien refactorizado, ya que se pretende que sea la última iteración.

- Añadir las últimas funcionalidades: permitir seleccionar una fila de la tabla resumen e resaltar el defecto al que pertenece y viceversa, es decir, seleccionar un defecto en la imagen y resaltar la fila a la que pertenece.
- Mejorar el margen de la selección al calcular umbrales locales.
- Permitir al usuario seleccionar más opciones.
- Permitir el cálculo de sólo las mejores características.
- Incluir un módulo de ayuda en línea, que el año pasado no existía.
- Solucionar algún BUG.

Además, se dan algunas ideas para líneas futuras, ya que a nosotros ya no nos da tiempo a meter más funcionalidad. Por ejemplo, se propone dar la opción al usuario de elegir él mismo qué características quiere calcular, automatizar la selección de las mejores características... También se propone comprobar qué tal funcionan las operaciones morfológicas de ImageJ, para ver si se puede incluir algo en nuestro proyecto o dejarlo como línea futura.

Se habla también de aspectos de la memoria, ya que tenemos que añadir unas breves descripciones de los algoritmos de clasificación utilizados y se determina que debemos incluir una extensa comparativa con el proyecto del año pasado en un anexo propio.

Por lo tanto, en este último *Sprint* pretendemos cerrar (o prácticamente dejarlo cerrado) toda la parte del código del proyecto, para dedicarnos a partir de aquí a finalizar la memoria y realizar una extensa comparativa con el año pasado (métricas, rendimiento...).

■ Fechas de desarrollo y tiempo estimado

Este *Sprint* se inicia el 18/04/2013 y tiene como fecha límite el 04/05/2013. Tiene un total de 22 puntos estimados.

■ Sprint planning

- S-01036: La aplicación debe permitir al usuario qué tipo de clasificación se va a usar: clases nominales o regresión.
  - TK-00078: Añadir la opción al fichero de propiedades, junto con su set y su get.
  - TK-00079: Añadir la opción al diálogo de opciones avanzadas.
- S-01037: La aplicación debe añadir un borde a la selección en aquellas zonas que no coincidan con los bordes de la imagen.
  - TK-00080: Modificar margen umbrales locales.
- S-01038: La aplicación debe permitir al usuario qué tipo de ventana defectuosa quiere usar.
  - TK-00081: Añadir la opción al fichero de opciones, junto con su set y su get.
  - TK-00082: Añadir la opción al diálogo de opciones avanzadas.
  - TK-00083: Añadir un Slider para seleccionar el porcentaje.

- S-01039: Se debe añadir un módulo de ayuda en línea (JavaHelp), accesible desde el menú y con teclas rápidas (F1).
  - TK-00084: Implementación del módulo de ayuda en línea.
  - TK-00085: Crear un buscador.
  - TK-00086: Implementar el acceso a la ayuda desde el menú y desde teclas rápidas (F1).
- S-01040: La aplicación debe poder permitir seleccionar una fila de la tabla de resultados e iluminar el defecto correspondiente.
  - TK-00087: Al seleccionar una fila, dibujar sólo el ROI al que pertenece.
  - TK-00088: Al seleccionar un defecto en la imagen, mostrar a qué fila de la tabla pertenece.
- S-01041: La aplicación debe dar la opción de elegir si se calculan todas las características o sólo las mejores.
  - TK-00089: Añadir la opción al fichero de propiedades, junto con su get y su set.
  - TK-00090: Añadir la opción al diálogo de opciones avanzadas.
  - TK-00091: En función de la opción, seleccionar todas las características o sólo las mejores.
- S-01042: La aplicación debe tratar las excepciones de forma correcta, mostrando avisos.
  - TK-00092: Capturar las excepciones de los hilos y almacenarlas.
  - TK-00093: Parar los hilos que aún estén en ejecución.
  - TK-00094: En el hilo de la interfaz comprobar si hay excepciones lanzadas. En caso positivo mostrar un mensaje con la excepción pertinente.
- B-01002: BUG: la aplicación debe impedir la reselección mientras se analiza.
  - TK-00095: Deshabilitar la selección de regiones mientras se analiza.

Burndown chart

En el *Burndown Chart* (ver figura A.8) podemos ver que nos adaptamos muy bien a la curva ideal en un principio, llegando incluso a adelantarnos y terminar antes de tiempo.

Retrospective meeting

¿Qué ha ido bien?

- La interacción con la tabla de resultados y los defectos dibujados ha salido bien y queda realmente bien.
- Las modificaciones realizadas funcionan bien.
- El módulo de ayuda en línea funciona perfectamente.

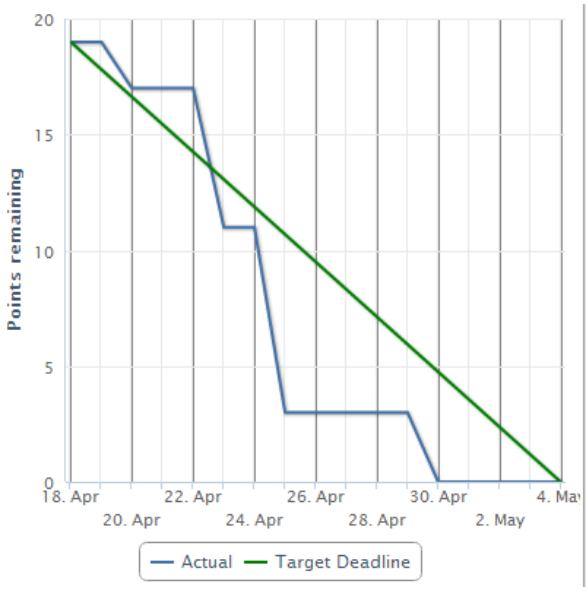

Figura A.8.: Burndown Chart del octavo Sprint

¿Qué dificultades ha habido?

- Por unas dudas que tuvimos, no pudimos implementar la selección de características, por lo que se deja para el siguiente *Sprint*. Es necesario, por tanto, hacer una iteración más.

A.5.9 Sprint 9

■ Planning meeting

En este *Sprint* lo que se pretende es:

- Terminar toda la parte del código definitivamente.
- Implementar lo que no se pudo implementar en el anterior *Sprint*
- Solucionar algún BUG.
- Añadir la última funcionalidad que surgió en la reunión anterior.

Como vemos, se pretende cerrar lo que no se pudo cerrar en el anterior *Sprint*. Será una iteración corta, con pocas historias de usuario.

La última funcionalidad hace referencia a la posibilidad de exportar la imagen con los defectos dibujados y la imagen con los defectos binarizados, así como la implementación de una medida de precisión, como es Precision & Recall.

Fechas de desarrollo y tiempo estimado

Este *Sprint* se inicia el 20/05/2013 y tiene como fecha límite el 03/06/2013. Tiene un total de 10 puntos estimados.

Sprint planning

- S-01041: La aplicación debe dar la opción de elegir si se calculan todas las características o sólo las mejores.
  - TK-00089: Añadir la opción al fichero de propiedades, junto con su get y su set.
  - TK-00090: Añadir la opción al diálogo de opciones avanzadas.
  - TK-00091: En función de la opción, seleccionar todas las características o sólo las mejores.
- S-01043: La aplicación debe permitir guardar la imagen final.
  - TK-00096: Creación del botón y su *listener*.
- S-01044: La aplicación debe permitir guardar los defectos binarizados.
  - TK-00097: Creación del botón y su *listener*.
- S-01045: La aplicación debe calcular «precision and recall» del resultado.
  - TK-00098: Creación de un botón y su *listener*.
  - TK-00099: Implementación de un diálogo modal con los datos.
  - TK-00100: Implementación del cálculo de «precision & recall».
- B-01003: BUG: coordenadas en lista de píxeles blancos.
  - TK-00101: Cambiar la forma en que se pasa la imagen a la estrategia.

Burndown chart

En el *Burndown Chart* (ver figura A.9) podemos ver que hay una gran parte de la iteración en la que la curva es plana. Esto es normal y lo tuvimos en cuenta al planificar, ya que teníamos los exámenes en esas semanas. Pese a ello, acabamos a tiempo.

Retrospective meeting

Aún no hay

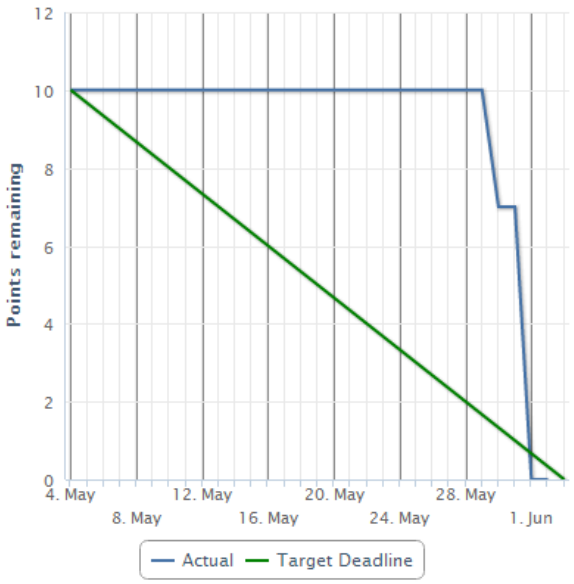

Figura A.9.: Burndown Chart del noveno Sprint

Universidad de Burgos

Escuela Politécnica Superior

Ingeniería Informática

Área de Lenguajes y Sistemas Informáticos

Anexo II - Especificación de requisitos

X-RayDetector: Detección automática de defectos en piezas metálicas mediante análisis de radiografías

Adrián González Duarte y Joaquín Bravo Panadero

Directores: Dr. César I. García Osorio
José Francisco Díez Pastor

B. ESPECIFICACIÓN DE REQUISITOS

B.1 Introducción

Este anexo tiene como objetivo analizar y documentar las necesidades funcionales que deberán ser soportadas por el sistema a desarrollar. Para ello, hay que identificar los requisitos que debe satisfacer el nuevo sistema, el estudio de los problemas de las unidades afectadas y sus necesidades actuales.

El objetivo de esta fase es describir lo que el sistema deberá ser capaz de hacer pero no cómo debe hacerlo.

Los tutores del proyecto toma un papel fundamental en este punto del proyecto, ya que asume el rol de cliente, es el encargado de especificar la funcionalidad que la aplicación debe implementar y establecer los plazos para lograrla.

Otra de las tareas a llevar a cabo en el presente anexo es la de priorizar cada una de las funcionalidades a implementar. Teniendo claras cuales son las tareas más críticas se podrá prestar un especial interés en ellas y asignar una cantidad mayor de recursos.

Hay que destacar la importancia de este documento, ya que sirve como punto de partida para el resto del trabajo. Por eso hay que hacer un esfuerzo extraordinario a la hora de definir y declarar correctamente todos los artefactos de ingeniería de software, para no tener que volver sobre nuestros pasos en etapas posteriores del trabajo. En primer lugar y de forma general, se recogen los objetivos que se quieren alcanzar con este trabajo. Estos objetivos son de muy alto nivel y estarán expresados por el cliente en lenguaje natural, siendo tarea del ingeniero su desglose y formalización. A continuación se detalla una lista de los usuarios que han participado en la toma de requisitos así como su relación con el sistema a construir.

Además, se incluye un catálogo con los requisitos que deberá cumplir el sistema a construir, junto con una breve explicación de cada uno, su tipo y su importancia. De estos requisitos se puede confeccionar la lista de los usuarios que interactúan con él, llamados actores. Se debe conocer el perfil de los usuarios para los que está destinada la aplicación, ya que esto condicionará en el futuro muchos aspectos del diseño, como por ejemplo las interfaces o la ayuda.

B.2 Objetivos del proyecto

El objetivo principal de este proyecto es el de mejorar la aplicación en diversos aspectos, entre los que se encuentran:

- Mejora del rendimiento, mediante inclusión de programación multihilo y sustitución de algunos cálculos por otros más eficientes.
- Mejora de la precisión en la detección de defectos, mediante la implementación de nuevos enfoques.
- Mejora de la interfaz gráfica, haciéndola más intuitiva y funcional.
- Mejora del diseño arquitectónico, enfocándolo hacia el mantenimiento y ampliación y solucionando fallos.
- Mejora del código.
- Implementación de nuevas funcionalidades, como el cálculo de características geométricas.

El proyecto, como vemos, está basado en uno anterior, pero se han tenido que cambiar un gran número de aspectos. Tenemos una parte muy orientada al mantenimiento, como es la mejora de todos los aspectos que ya existen, y otra de adición de nuevas funcionalidades.

Las nuevas funcionalidades incluyen la detección de defectos usando nuevos enfoques, es decir, mediante el uso de los filtros de umbrales locales y nuevas formas de determinar cuándo una ventana es defecto, así como una detección de defectos más interactiva, permitiendo al usuario interactuar con los defectos dibujados. También se incluye la implementación del cálculo de características geométricas y la inclusión de estos cálculos en una tabla interactiva. Estos cálculos permitirán, de una forma sencilla, incluir una clasificación de los defectos en tipos.

B.3 Usuarios participantes

Durante la fase de análisis han participado diversos usuarios, cada uno desempeñando uno o varios papeles. De este modo han sido establecidos los requisitos necesarios para el desarrollo del proyecto software. Cabe destacar la participación de los tutores de este proyecto.

- José Francisco Díez Pastor y César Ignacio García Osorio, tutores del proyecto, han asumido diversos roles durante el proceso:
  - Como *cliente*, han participado en la fase de análisis describiendo las funcionalidades y el comportamiento del sistema a desarrollar. Asimismo, han organizado el calendario y marcado los plazos de entrega.
  - Como técnicos, ofreciendo sus conocimientos sobre la minería de datos. Además, César ha aportado sus conocimientos de LaTeX y experiencia en la realización de proyectos previos, y José su experiencia con la biblioteca Weka y con la programación en Java y sus conocimientos en las técnicas de procesamiento digital de imagen.
- Por último, Adrián González Duarte y Joaquín Bravo Panadero han asumido el rol de *analista*. Este rol agrupa las siguientes responsabilidades: analizar y describir el problema planteado por el cliente y realizar el diseño con la solución propuesta.

B.4 Descripción del sistema actual

En este apartado se pretende hacer referencia a la aplicación en la que hemos integrado las funcionalidades descritas anteriormente.

El proyecto en el que está basada nuestra herramienta es **X-Ray Detector: Detección de defectos en piezas metálicas mediante análisis de radiografías**, por Alan Blanco Álamo y Víctor Barbero García, junio de 2012. Este proyecto incluía la funcionalidad de entrenamiento, detección y dibujado de defectos que tiene nuestros proyecto. En un principio, se pensó en simplemente refactorizar su código e ir incluyendo sobre él las nuevas modificaciones, pero al poco tiempo se vio que esto no era del todo sencillo, ya que el diseño y el propio código no eran de la calidad esperada (diseño sin ningún tipo de patrón que facilite la inclusión de nuevas funcionalidades, defectos de código o *bad smells*...), por lo que decidimos rediseñar la aplicación desde el principio prácticamente.

Se ha aprovechado todo lo que se ha podido del proyecto anterior, como son los cálculos de características, si bien se han cambiado un poco para permitir, por ejemplo, que puedan ser usadas a modo de librería. También se ha mejorado alguno de estos cálculos, bien usando funciones e ImageJ o bien usando otras librerías, como EJML.

Una vez rediseñada la aplicación, se ha procedido a incluir las nuevas funcionalidades o mejoras poco a poco sobre este nuevo diseño.

B.5 Factores de riesgo

En este apartado se analizan las dificultades a las que se va a tener que enfrentar el desarrollo de este proyecto software. Identificar los riesgos servirá para estar alerta sobre los posibles problemas que puedan surgir y los retrasos que éstos desencadenen.

Los factores de riesgo que han sido identificados son:

- Falta de conocimiento: la primera dificultad encontrada es el desconocimiento de los fundamentos teóricos sobre temas como análiss de imágenes, etc.
- Complejidad de la documentación existente: la documentación disponible se basa en los artículos sobre los algoritmos a implementar y en algún que otro artículo de investigación, por lo que la labor de comprensión de información se presenta difícil.
- Complejidad de la aplicación del proyecto del año pasado: se hace difícil comprender el proyecto del año pasado, sobre todo en cuanto al código.
- Desconocimiento sobre herramientas: nunca habíamos trabajado con una herramienta como ImageJ, o con librerías complejas como EJML, lo que hace complicado su uso.
- Desconocimiento sobre la metodología de desarrollo: si bien es cierto que teníamos una buena base teórica al respecto, nunca habíamos podido poner en práctica los conocimientos sobre las metodologías ágiles, como Scrum, lo que hace algo complicado crear un *backlog*, usar un *tracker*, etc.
- Nuevo sistema de composición de texto: para la documentación se va a utilizar LaTeX lo que provocará una carga extra al desarrollo de la memoria del proyecto. Se utilizará para lograr una apariencia más sólida y profesional.

Se hace patente que para la creación de la aplicación se requerirá de una gran labor de investigación y análisis que posibilite superar todos los riesgos identificados y que concluyan con la superación de los objetivos marcados en los plazos establecidos.

B.6 Catálogo de requisitos del sistema

El objetivo de este apartado es definir de forma clara, precisa, completa y verificable todas las funcionalidades y restricciones del sistema a construir.

B.6.1 Objetivos del proyecto

Los objetivos salen, en parte, de los ya descritos en el apartado B.2 y, por otra parte, de los objetivos de la aplicación del año pasado. A continuación, se especifican los objetivos en la correspondiente plantilla de objetivos:

OBJ - 01	**Abrir imagen**
Descripción	La aplicación debe ser capaz de abrir y mostrar una imagen en la aplicación.
Autores	Adrián González Duarte Joaquín Bravo Panadero
Importancia	Vital
Urgencia	Alta
Comentarios	Es uno de los objetivos principales de la aplicación y, por tanto, un objetivo general

OBJ - 02	**Entrenar Clasificador**
Descripción	La aplicación debe ser capaz de entrenar un clasificador para realizar posteriormente el proceso de detección de defectos.
Autores	Adrián González Duarte Joaquín Bravo Panadero
Importancia	Vital
Urgencia	Alta
Comentarios	Es uno de los objetivos principales de la aplicación y, por tanto, un objetivo general

OBJ - 03	**Analizar Imagen**
Descripción	La aplicación debe ser capaz de analizar imágenes y detectar defectos si los tuviera.
Autores	Adrián González Duarte Joaquín Bravo Panadero
Importancia	Vital
Urgencia	Alta
Comentarios	Es uno de los objetivos principales de la aplicación y, por tanto, un objetivo general

continúa en la página siguiente

continúa desde la página anterior

OBJ - 04	**Exportar Resultados**
Descripción	La aplicación debe ser capaz de exportar los resultados de las operaciones que ha ido realizando la aplicación a un log.
Autores	Adrián González Duarte Joaquín Bravo Panadero
Importancia	Alta
Urgencia	Media
Comentarios	El formato del informe a exportar será en HTML.

OBJ - 05	**Mostrar Características y Seleccionar Defecto**
Descripción	La aplicación debe ser capaz de mostrar las características de un defecto e identificarlo seleccionándolo en una tabla de resultados o directamente sobre la imagen analizada.
Autores	Adrián González Duarte Joaquín Bravo Panadero
Importancia	Alta
Urgencia	Media
Comentarios	Nos referimos a las características geométricas.

Tabla B.1.: Objetivos

B.6.2 Descripción de los actores

A continuación, se describen los actores, es decir, los usuarios externos que interactúan con el sistema.

ACT - 01	**Usuario**
Versión	1.0
Autores	Adrián González Duarte Joaquín Bravo Panadero
Descripción	Este actor representa al usuario que quiere utilizar la aplicación.
Comentarios	Persona física que interactúa con el sistema y realiza las operaciones pertinentes de entrenamiento de clasificadores y análisis de imágenes.

Tabla B.2.: Actores

B.6.3 Funciones del producto

Una de las características más importantes de la aplicación que pretende desarrollarse es la facilidad de uso. Dado que su empleo está orientado a convertirse en una herramienta empresarial que ayude a los operarios a identificar piezas defectuosas, debe ser fácil de usar y tener la versatilidad

para poder entrenar nuevos conjuntos de imágenes de cualquier nueva pieza que pueda desarrollar la empresa.

La aplicación cuenta con un panel de Log que muestra al usuario datos sobre el progreso del proceso que se está ejecutando.

Para cargar y guardar los conjuntos de datos se usará el formato nativo de *Weka* (*ARFF*).

■ Características a calcular

La aplicación deberá calcular las siguientes características, tanto para la imagen normal como para la imagen con Saliency Map aplicado.

- Características estándar:
  1. Media
  2. Desviación estándar
  3. Primera derivada
  4. Segunda derivada
- Características de Haralick:
  1. Segundo Momento Angular
  2. Contraste
  3. Correlación
  4. Suma de cuadrados
  5. Momento Diferencial Inveros
  6. Suma Promedio
  7. Suma de Entropías
  8. Suma de Varianzas
  9. Entropía
  10. Diferencia de Varianzas
  11. Diferencia de Entropías
  12. Medidas de Información de Correlación 1
  13. Medidas de Información de Correlación 2
  14. Coeficiente de Correlación
- Local Binary Patterns

Además, debe ser capaz de calcular las siguientes características geométricas sobre los defectos encontrados:

- Área

- Perímetro
- Circularidad
- Redondez
- Semieje mayor
- Semieje menor
- Ángulo
- Distancia Feret

■ Entrenar el clasificador

Esta parte deberá posibilitar las siguientes funciones:

- Se permitirá al usuario abrir un directorio que contenga radiografías etiquetadas y listas para entrenar.
- Se informará al usuario en caso de que el directorio no sea apropiado, ya sea porque el número de imágenes originales y máscaras no coinciden, sus nombres no son los mismos, etc.
- Una vez cargado el directorio, se podrá crear un fichero *ARFF* que contendrá el cálculo de todas las características para las imágenes de dicho directorio. Este fichero podrá ser guardado donde elija el usuario.
- El usuario podrá decidir si quiere que la extracción de características se haga mediante regiones aleatorias o con una ventana deslizante que recorra toda la imagen.
- Durante el cálculo de características, se informará al usuario del progreso y de los pasos que se van ejecutando mediante un Log, que podrá ser guardado en el disco duro. Asimismo, una barra de progreso le indicará el porcentaje aproximado de proceso ejecutado.
- La ejecución del cálculo de características podrá ser detenida en cualquier momento a petición del usuario.
- El entrenamiento se puede realizar también con un *ARFF* generado previamente.

■ Usar el clasificador para detectar defectos

En este apartado se incluyen las siguientes funciones:

- El usuario podrá abrir una imagen para detectar sus defectos.
- Se podrá iniciar la detección de defectos, teniendo que cargar un modelo entrenado.
- Se podrá seleccionar un área de la imagen para detectar los defectos contenidos en él. Si no se selecciona un área, se debe avisar al usuario de que el proceso puede llevar mucho tiempo.
- La detección de defectos podrá ser detenida en cualquier momento a petición del usuario.

- Se mostrará la imagen con los defectos marcados en un panel y la información de las características geométricas en una tabla.
- Se podrá seleccionar un defecto en la imagen para identificar la fila de la tabla de características geométricas que contiene sus datos. La inversa, seleccionar una fila y que se ilumine el defecto, también debe ser posible.

B.6.4 Requisitos de usuario

El usuario debe ser capaz de utilizar la aplicación mediante la interfaz gráfica cargando imágenes, detectando defectos y visualizando los resultados.

La ayuda será un punto a tener en cuenta y en todo momento el usuario podrá solicitarla para orientarse y resolver dudas que le puedan surgir durante el manejo de la misma.

B.6.5 Requisitos de sistema

Al tratarse de una aplicación de escritorio será necesario contar con un hardware que tenga unos requisitos mínimos.

La aplicación se va a desarrollar en *Java* por lo que no debe ser muy exigente, es decir, cualquier máquina capaz de hacer correr la máquina virtual de *Java* debe ser capaz de ejecutar la aplicación.

No obstante, dependiendo del tamaño del fichero *ARFF* que se vaya a cargar, o del número de características que se quieran calcular, será recomendable una mayor capacidad hardware para reducir el tiempo de ejecución.

Los requisitos mínimos serán:

- Equivalente a 2.8 GHz.
- 2 GB de memoria RAM.
- Resolución de pantalla igual o superior a 1200 x 720

Mientras que los requisitos recomendados serán:

- Equivalente a 2.8 GHz(doble núcleo).
- 4 GB de memoria RAM.
- Resolución de pantalla igual o superior a 1280 x 960

Aunque estos requisitos dependerán en gran medida de el número de instancias o características con el que se trabaje.

En cuanto al sistema operativo, deberá ser posible la ejecución en todos los que exista una versión de máquina virtual de *Java*. Uno de los beneficios de que el proyecto se implemente en un lenguaje interpretado es que maximiza las posibilidades de uso en distintos entornos.

La versión de *Java* sobre la que se desarrollará el proyecto es la 1.7 (o 7 directamente).

B.7 Especificación de los requisitos

Esta sección del anexo recoge los casos de uso que representan la funcionalidad que debe cubrir la aplicación a desarrollar. Cada caso de uso es una descripción de una secuencia de acciones que se ejecutan en un sistema para producir la salida esperada por un actor.

El primer lugar quedan definidos los requisitos inherentes a la tecnología utilizada y después los requisitos de información. Con este primer análisis se detallan a continuación los requisitos funcionales, apoyados en los casos de uso, y por último los requisitos no funcionales.

B.7.1 Especificación de requisitos inherentes a la tecnología utilizada

Son los requisitos que vienen «heredados» por la tecnología a utilizar.

Se ha elegido el lenguaje de programación *Java* para el desarrollo del proyecto, concretamente la versión 7. El requisito necesario para poder ejecutar la aplicación será que el terminal tenga instalada una máquina virtual de *Java*.

B.7.2 Especificación de requisitos de información

El objetivo de este apartado es determinar la información que se debe almacenar para cumplir los objetivos anteriormente descritos y que el programa funcione. Con objeto de identificar cada uno de los requisitos de información se les ha asignado un código único y un nombre descriptivo. A continuación (ver tabla B.3) se detallan de manera tabulada cada uno de los requisitos en plantillas.

RI - 01	**Información cálculo de características**
Descripción	Almacena la información resultante del cálculo de características de una radiografía mediante ficheros *ARFF*
Datos específicos	Características calculadas a partir de una radiografía
Importancia	Alta
Comentarios	Esta información se utilizará para entrenar un clasificador

RI - 02	**Información de configuración de la ventana de análisis**
Descripción	Almacena la información necesaria para analizar la imagen mediante ventanas
Datos específicos	Tamaño de la ventana
	Salto entre ventanas
	Tipo de ventana (aleatoria, deslizante)
	Cuándo una ventana es defectuosa (entrenamiento)
Importancia	Alta
Comentarios	Se almacena todo en un mismo fichero de configuración

continúa en la página siguiente

continúa desde la página anterior

RI - 03	**Información para el entrenamiento del clasificador o para el dibujado de características**
Descripción	Almacena la información necesaria para entrenar un clasificador o para dibujar características
Datos específicos	Fichero *ARFF* generado al calcular las características de una radiografía
Importancia	Alta
Comentarios	Es obligatorio para poder entrenar al clasificador

RI - 04	**Información para la detección de defectos**
Descripción	Almacena la información necesaria detectar los defectos de una radiografía
Datos específicos	Modelo entrenado que utilizará la aplicación para detectar los defectos de una radiografía
Importancia	Alta
Comentarios	Es obligatorio para poder detectar los defectos

Tabla B.3.: Requisitos de información

B.7.3 Requisitos funcionales

La definición de los requisitos funcionales servirá de ayuda para el diseño de la herramienta.

En la lista aparecen todos los requisitos identificados en la aplicación:

- Cargar imágenes: el usuario podrá cargar imágenes para detectar defectos.
- Detención del proceso: se podrá detener el proceso en cualquier momento.
- Cargar instancias: la aplicación debe estar capacitada para abrir conjuntos de instancias en el formato *ARFF* soportado por *Weka*.
- Entrenar clasificador: se podrá entrenar un clasificador para detectar los defectos de nuevas radiografías, bien sea con un *ARFF* ya creado o creando uno nuevo.
- Detectar defectos: se podrá realizar la detección de defectos de cualquier imagen que el usuario cargue, pudiendo seleccionar la región exacta de la imagen que se quiera analizar.
- Visualización de resultados: se podrán visualizar los defectos detectados, así como una tabla con las características geométricas.
- Interacción con los resultados: se podrá seleccionar un defecto y se iluminará la fila de la tabla correspondiente a ese defecto. Se podrá seleccionar también una fila de la tabla y se iluminará el defecto.
- Guardar Log: el usuario podrá visualizar el progreso en un log y guardarlo en el disco duro.
- Visualización de la ayuda: las interfaces deberán ofrecer la ayuda al usuario que necesite para poder utilizarlas.

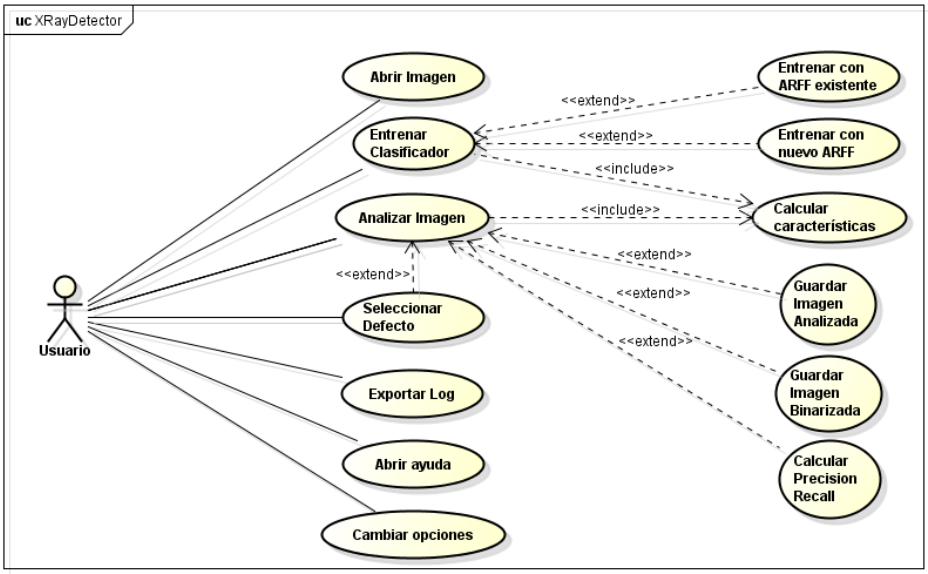

Figura B.1.: Diagrama de casos de uso principal

- Guardar imagen analizada: se permite guardar los resultados del análisis en un fichero.
- Guardar imagen binarizada: se permite guardar una binarización de los defectos encontrados.
- Calcular «Precision & Recall»: se realiza el cálculo de estas medidas de precisión, a través de una máscara.

Para la definición formal de los casos de uso se hace uso de los diagramas de casos de uso. Para completar la información de los diagramas se utilizan las plantillas donde quedan reflejados los aspectos funcionales.

En el primer diagrama (ver figura B.1) se muestra el diagrama de casos de uso principal.

RF-01 Abrir imagen		
Versión	1.0	
Autores	Adrián González Duarte Joaquín Bravo Panadero	
Objetivos asociados	OBJ-01	
Descripción	Permite abrir una imagen en la aplicación y mostrarla al usuario para posteriormente ser analizada en busca de posibles defectos	
Precondiciones	Debe existir la imagen en el sistema	
Secuencia normal	Paso	Acción
	1	Seleccionar la opción de abrir imagen
	2	El sistema pedirá al usuario que especifique que imagen quiere cargar en la aplicación a partir de un explorador de archivos
	3	Se carga la imagen en la aplicación
Postcondiciones	Imagen importada en la aplicación	
Excepciones	Paso	Acción
	1	Si la operación se cancela el caso de uso finaliza
Rendimiento		
Frecuencia	Alta, ~~Media, Baja~~	
Importancia	Alta, ~~Media, Baja~~	
Urgencia	Alta, ~~Media, Baja~~	
Comentarios	Va a permitir realizar más acciones, como analizar	

Tabla B.4.: Caso de uso: RF-01 Abrir imagen

A continuación, aparecen los casos de uso listados y cada una de las plantillas:

- RF-01 Abrir imagen (ver tabla B.4)
- RF-02 Entrenar clasificador (ver tabla B.5)
- RF-03 Entrenar clasificador con ARFF existente (ver tabla B.6)
- RF-04 Entrenar clasificador generando nuevo ARFF (ver tabla B.7)
- RF-05 Analizar imagen (ver tabla B.8)
- RF-06 Calcular características (ver tabla B.9)
- RF-07 Seleccionar defecto (ver tabla B.10)
- RF-08 Exportar Log (ver tabla B.11)
- RF-09 Abrir ayuda (ver tabla B.12)
- RF-10 Cambiar opciones (ver tabla B.13)
- RF-11 Guardar imagen analizada (ver tabla B.14)
- RF-12 Guardar imagen binarizada (ver tabla B.15)
- RF-13 Calcular «Precision & Recall» (ver tabla B.16)

RF-02 Entrenar clasificador		
Versión	1.0	
Autores	Adrián González Duarte Joaquín Bravo Panadero	
Objetivos asociados	OBJ-02	
Descripción	Permite a la aplicación entrenar un clasificador a partir de un conjunto de imágenes o de un archivo ARFF existente	
Precondiciones	Ninguna	
Secuencia normal	Paso	Acción
	1	El usuario selecciona la opción de entrenar clasificador
	2	El sistema pedirá al usuario cómo se va a entrenar el clasificador, mediante un ARFF existente o mediante un conjunto de imágenes
	3	El sistema entrena un clasificador
Postcondiciones	El sistema entrena un clasificador	
Excepciones	Paso	Acción
	2	Si la operación se cancela en el proceso de selección del tipo de entrenamiento el caso de uso finaliza
	3	Si el proceso se para a lo largo de su ejecución el caso de uso finaliza
Rendimiento		
Frecuencia	~~Alta,~~ Media, ~~Baja~~	
Importancia	Alta, ~~Media, Baja~~	
Urgencia	Alta, ~~Media, Baja~~	
Comentarios	Es una característica imprescindible, ya que sin un clasificador la aplicación no puede detectar defectos en una imagen posteriormente	

Tabla B.5.: Caso de uso: RF-02 Entrenar clasificador

RF-03 Entrenar clasificador con ARFF existente		
Versión	1.0	
Autores	Adrián González Duarte Joaquín Bravo Panadero	
Objetivos asociados	OBJ-02	
Descripción	Permite a la aplicación entrenar un clasificador a partir de un archivo ARFF existente	
Precondiciones	Debe existir un conjunto de imágenes en el sistema junto con su conjunto de máscaras	
Secuencia normal	Paso	Acción
	1	Seleccionar la opción de entrenar clasificador. Y el usuario especifica que quiere entrenar un clasificador a partir de un archivo ARFF existente
	2	El sistema pedirá al usuario que especifique la ruta del archivo ARFF con el que se desea realizar el proceso de entrenamiento
	3	El sistema entrena un clasificador
Postcondiciones	Clasificador entrenado	
Excepciones	Paso	Acción
	2	Si la operación se cancela antes de seleccionar que se desea utilizar un ARFF existente el caso de uso finaliza
	3	Si el proceso se para a lo largo de su ejecución el caso de uso finaliza
Rendimiento		
Frecuencia	~~Alta,~~ Media, ~~Baja~~	
Importancia	Alta, ~~Media, Baja~~	
Urgencia	Alta, ~~Media, Baja~~	
Comentarios	Es una característica imprescindible, ya que sin un clasificador la aplicación no puede detectar defectos en una imagen posteriormente	

Tabla B.6.: Caso de uso: RF-03 Entrenar clasificador con ARFF existente

RF-04 Entrenar clasificador generando nuevo ARFF		
Versión	1.0	
Autores	Adrián González Duarte Joaquín Bravo Panadero	
Objetivos asociados	OBJ-02	
Descripción	Permite a la aplicación entrenar un clasificador a partir de un conjunto de imágenes	
Precondiciones	Debe existir un conjunto de imágenes en el sistema junto con su conjunto de máscaras	
Secuencia normal	Paso	Acción
	1	Seleccionar la opción de entrenar clasificador. El usuario especificará que desea entrenar un clasificador a partir de un conjunto de imágenes
	2	El sistema pedirá al usuario que especifique la carpeta que contiene las imágenes con las que se va a entrenar el clasificador
	3	El sistema inicia un proceso de extracción de características sobre cada una de las imágenes generando un archivo ARFF nuevo. Se inicia el caso de uso RF-06 Calcular características
	4	El sistema entrena un clasificador
Postcondiciones	Clasificador entrenado	
Excepciones	Paso	Acción
	2	Si la operación se cancela antes de seleccionar que se desea generar un ARFF nuevo el caso de uso finaliza
	2	Si el proceso se para a lo largo de su ejecución el caso de uso finaliza
Rendimiento		
Frecuencia	~~Alta,~~ Media, ~~Baja~~	
Importancia	Alta, ~~Media, Baja~~	
Urgencia	Alta, ~~Media, Baja~~	
Comentarios	Es una característica imprescindible, ya que sin un clasificador la aplicación no puede detectar defectos en una imagen posteriormente	

Tabla B.7.: Caso de uso: RF-04 Entrenar clasificador generando nuevo ARFF

RF-05 Analizar imagen		
Versión	1.0	
Autores	Adrián González Duarte Joaquín Bravo Panadero	
Objetivos asociados	OBJ-03	
Descripción	Permite a la aplicación analizar una imagen en busca de posibles defectos	
Precondiciones	Debe existir un clasificador entrenado así como una imagen importada en la aplicación	
Secuencia normal	Paso	Acción
	1	Seleccionar la opción de entrenar analizar imagen y elección de clasificador
	2	El sistema inicia el proceso de análisis de la imagen en busca de defectos. Se inicia en caso de uso RF-06 Calcular características
	3	La aplicación muestra los resultados del análisis
Postcondiciones	Imagen analizada: defectos dibujados y características geométricas listadas	
Excepciones	Paso	Acción
	1	Si se cancelar la elección de clasificador, el casi de uso finaliza
	2	Si el proceso se para a lo largo de su ejecución el caso de uso finaliza
Rendimiento		
Frecuencia	Alta, ~~Media, Baja~~	
Importancia	Alta, ~~Media, Baja~~	
Urgencia	Alta, ~~Media, Baja~~	
Comentarios	Es una característica imprescindible, ya que es el objetivo principal del proyecto	

Tabla B.8.: Caso de uso: RF-05 Analizar imagen

RF-06 Calcular características		
Versión	1.0	
Autores	Adrián González Duarte Joaquín Bravo Panadero	
Objetivos asociados	OBJ-02, OBJ-03	
Descripción	Calcular las características de uma imagen	
Precondiciones	Debe haberse iniciado el caso de uso RF-04 o el RF-05	
Secuencia normal	Paso	Acción
	1	El sistema reciba una región a analizar
	2	El sistema calcula las características (todas o las mejores) sobre la región
	3	Si se había iniciado el caso de uso RF-04, se generará una nueva línea en el ARFF. Si era el RF-05, se pasarán las características al clasificador, que dirá si corresponden con un defecto o no
Postcondiciones	Características calculadas	
Excepciones	Paso	Acción
	2	Si hay algún problema al calcular, el sistema mostrará un error y el caso de uso finaliza
Rendimiento		
Frecuencia	Alta, ~~Media, Baja~~	
Importancia	Alta, ~~Media, Baja~~	
Urgencia	Alta, ~~Media, Baja~~	
Comentarios	Es una característica básica, pues forma parte tanto del entrenamiento de clasificadores como de la detección de defectos	

Tabla B.9.: Caso de uso: RF-06 Calcular características

RF-07 Seleccionar defecto		
Versión	1.0	
Autores	Adrián González Duarte Joaquín Bravo Panadero	
Objetivos asociados	OBJ-05	
Descripción	Muestra al usuario los defectos detectados en una imagen tras ser analizada con sus características. Permite seleccionar un defecto dibujado para ver sus características.	
Precondiciones	Debe haber finalizado el caso de uso RF-05	
Secuencia normal	Paso	Acción
	1	Se ejecuta el caso de uso RF-05
	2	La aplicación muestra los resultados del análisis indicando los defectos que ha encontrado junto con sus características en una tabla de resultados
	3	El usuario selecciona bien el defecto o bien unos resultados en la tabla y tanto el defecto como sus características asociadas se seleccionan para ser mejor identificados
Postcondiciones	Defectos detectados y seleccionados y características calculadas	
Excepciones	Paso	Acción
Rendimiento		
Frecuencia	Alta, ~~Media, Baja~~	
Importancia	Alta, ~~Media, Baja~~	
Urgencia	~~Alta,~~ Media, ~~Baja~~	
Comentarios	En este caso de uso, nos estamos refiriendo a las características geométricas. Proporciona la interactividad con los resultados.	

Tabla B.10.: Caso de uso: RF-07 Seleccionar defecto

RF-08 Exportar Log		
Versión	1.0	
Autores	Adrián González Duarte Joaquín Bravo Panadero	
Objetivos asociados	OBJ-04	
Descripción	Permite al usuario exportar un log con los resultados de los procesos ejecutados a lo largo de toda la aplicación	
Precondiciones	Ninguna	
Secuencia normal	Paso	Acción
	1	El usuario pulsa el botón «Exportar Log»
	2	El sistema genera un log en formato HTML que mostrará todos los procesos llevados a cabo por la aplicación junto con sus resultados
Postcondiciones	Log exportado	
Excepciones	Paso	Acción
Rendimiento		
Frecuencia	~~Alta, Media,~~ Baja	
Importancia	~~Alta,~~ Media, ~~Baja~~	
Urgencia	~~Alta, Media,~~ Baja	
Comentarios		

Tabla B.11.: Caso de uso: RF-08 Exportar Log

RF-09 Abrir ayuda		
Versión	1.0	
Autores	Adrián González Duarte Joaquín Bravo Panadero	
Objetivos asociados		
Descripción	Permite visualizar la ayuda de la aplicación	
Precondiciones	Ninguna	
Secuencia normal	Paso	Acción
	1	El usuario abre la ayuda, bien mediante una tecla especial o bien mediante una opción en un menú
	2	Se mostrará la ayuda
Postcondiciones	Se mostrará una nueva ventana en la que se encuentra la ayuda	
Excepciones	Paso	Acción
Rendimiento		
Frecuencia	~~Alta,~~ Media, ~~Baja~~	
Importancia	~~Alta,~~ Media, ~~Baja~~	
Urgencia	~~Alta, Media,~~ Baja	
Comentarios		

Tabla B.12.: Caso de uso: RF-09 Abrir ayuda

B.7.4 Requisitos no funcionales

Una vez analizados los requisitos de información y los funcionales falta un tipo de requisitos que, normalmente, suelen ser de carácter técnico y se engloban dentro de requisitos no funcionales.

A continuación aparecen listados los requisitos no funcionales a tener en cuenta para el diseño de la aplicación:

- **Extensible:** debe estar pensado para que se pueda ampliar añadiendo nuevas características a calcular, utilizando la misma interfaz.
- **Facilidad de uso de la interfaz:** la aplicación debe tener una interfaz que sea fácil de manejar y de entender por cualquier tipo de usuario. Debe ser intuitiva.
- **Ayuda:** la aplicación debe tener explicaciones de ayuda para los usuarios en las partes que puedan ser más difíciles de entender o manejar, o como aclaración de algún concepto.
- **Documentado:** si se pretende que un software sea ampliado por terceros, o continuado en un futuro, éste debe estar debidamente documentado.

RF-10 Cambiar opciones		
Versión	1.0	
Autores	Adrián González Duarte Joaquín Bravo Panadero	
Objetivos asociados	OBJ-02, OBJ-03	
Descripción	Permite cambiar algunas opciones de la aplicación que afectarán a los procesos de entrenamiento y detección	
Precondiciones	Ninguna	
Secuencia normal	Paso	Acción
	1	El usuario selecciona «opciones avanzadas»
	2	El usuario determina cuáles de las opciones quiere cambiar y con qué valores
	3	El sistema guarda los cambios en un fichero de propiedades
Postcondiciones	El fichero de propiedades contiene los nuevos cambios	
Excepciones	Paso	Acción
	3	Si se cancela el proceso, los cambios no se guardan
Rendimiento		
Frecuencia	~~Alta,~~ Media, ~~Baja~~	
Importancia	Alta, ~~Media, Baja~~	
Urgencia	~~Alta,~~ Media, ~~Baja~~	
Comentarios	Las opciones que se pueden cambiar están relacionadas con el tamaño de las ventanas y de su salto, el tipo de heurística de ventana defectuosa, el tipo de ventana para entrenar, el tipo de detección...	

Tabla B.13.: Caso de uso: RF-10 Cambiar opciones

RF-11 Guardar imagen analizada		
Versión	1.0	
Autores	Adrián González Duarte Joaquín Bravo Panadero	
Objetivos asociados	OBJ-03	
Descripción	Permite exportar la imagen analizada junto con los defectos dibujados a un fichero	
Precondiciones	Debe haber finalizado el caso de uso RF-05	
Secuencia normal	Paso	Acción
	1	El usuario selecciona «guardar imagen»
	2	El usuario determina el fichero en el que guardar la imagen
	3	El sistema guarda la imagen en el fichero especificado
Postcondiciones	Se ha creado un fichero con la imagen	
Excepciones	Paso	Acción
	2	Si se cancela el proceso, la imagen no se guarda
Rendimiento		
Frecuencia	~~Alta, Media,~~ Baja	
Importancia	~~Alta, Media,~~ Baja	
Urgencia	~~Alta, Media,~~ Baja	
Comentarios	La imagen se guardará en un fichero con formato JPG	

Tabla B.14.: Caso de uso: RF-11 Guardar imagen analizada

RF-12 Guardar imagen binarizada		
Versión	1.0	
Autores	Adrián González Duarte Joaquín Bravo Panadero	
Objetivos asociados	OBJ-03	
Descripción	Permite exportar los defectos detectados en una imagen binarizada a un fichero	
Precondiciones	Debe haber finalizado el caso de uso RF-05	
Secuencia normal	Paso	Acción
	1	El usuario selecciona «guardar defectos binarizados»
	2	El usuario determina el fichero en el que guardar la imagen
	3	El sistema guarda la imagen en el fichero especificado
Postcondiciones	Se ha creado un fichero con la imagen	
Excepciones	Paso	Acción
	2	Si se cancela el proceso, la imagen no se guarda
Rendimiento		
Frecuencia	~~Alta, Media,~~ Baja	
Importancia	~~Alta, Media,~~ Baja	
Urgencia	~~Alta, Media,~~ Baja	
Comentarios	La imagen se guardará en un fichero con formato JPG	

Tabla B.15.: Caso de uso: RF-12 Guardar imagen binarizada

RF-13 Calcular «Precision & Recall»		
Versión	1.0	
Autores	Adrián González Duarte Joaquín Bravo Panadero	
Objetivos asociados	OBJ-03	
Descripción	Permite realizar los cálculos de precision & recall sobre los defectos localizados	
Precondiciones	Debe haber finalizado el caso de uso RF-05	
Secuencia normal	Paso	Acción
	1	El usuario selecciona «calcular precision & recall»
	2	El usuario determina cuál es la máscara asociada a la imagen
	3	El sistema realiza los cálculos y los muestra
Postcondiciones	Se muestra un diálogo con los datos	
Excepciones	Paso	Acción
	2	Si se cancela el proceso, el caso de uso finaliza
Rendimiento		
Frecuencia	~~Alta, Media,~~ Baja	
Importancia	~~Alta,~~ Media, ~~Baja~~	
Urgencia	~~Alta,~~ Media, ~~Baja~~	
Comentarios	Debe existir una máscara sobre la cual se van a realizar las comparaciones	

Tabla B.16.: Caso de uso: RF-13 Calcular «Precision & Recall»

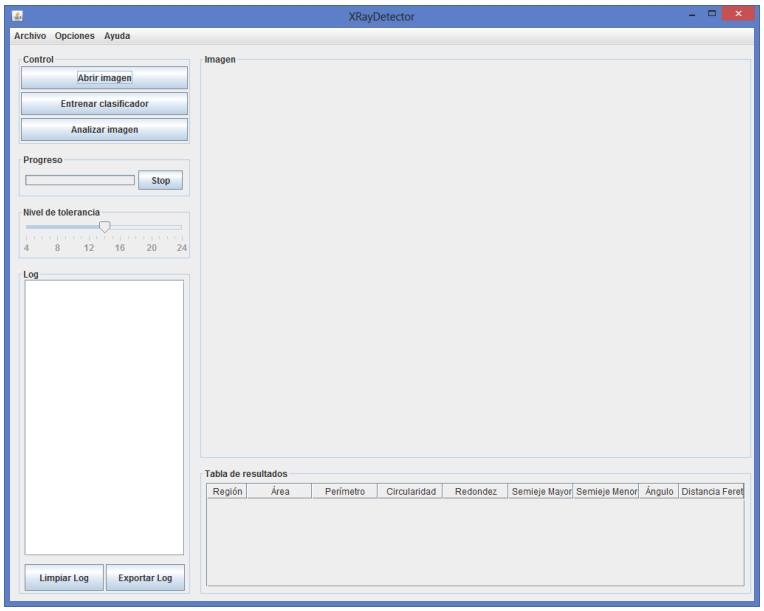

Figura B.2.: Prototipo de la ventana principal

B.8 Interfaz de usuario

En este apartado se muestran una serie de ventanas correspondientes a la interfaz de usuario con la que la aplicación es capaz de cumplir los requisitos funcionales que se han establecido anteriormente.

Debido a que estamos aún en la parte de análisis del proyecto, estas interfaces tan solo son bocetos de las interfaces que tendrá la aplicación definitiva.

En la imagen (ver figura B.2) se muestra el prototipo de la ventana principal, desde la que se pueden realizar todas las funcionalidades.

En la imagen (ver figura B.3) se muestra el prototipo de la ventana de opciones avanzadas, desde la que podemos cambiar un cierto número de opciones.

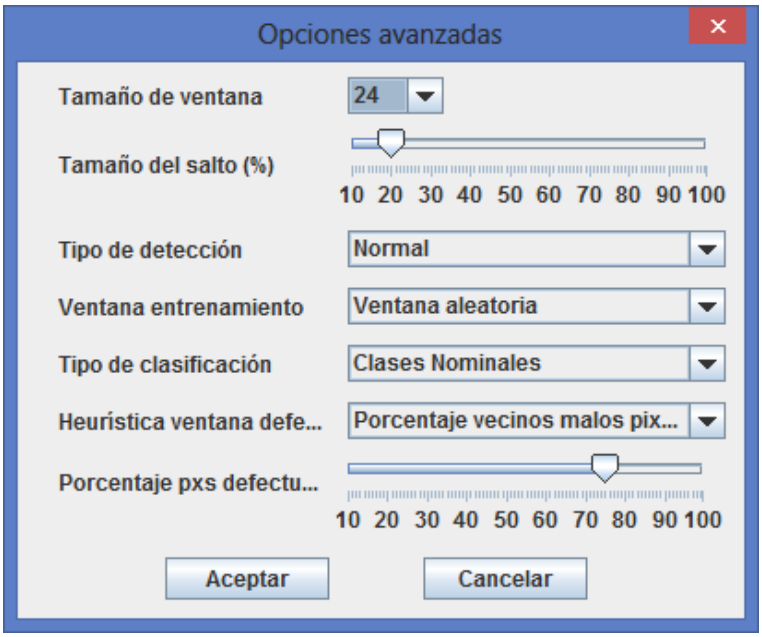

Figura B.3.: Prototipo de la ventana de opciones avanzadas

Universidad de Burgos

Escuela Politécnica Superior

Ingeniería Informática

Área de Lenguajes y Sistemas Informáticos

Anexo III - Especificación de diseño

X-RayDetector: Detección automática de defectos en piezas metálicas mediante análisis de radiografías

Adrián González Duarte y Joaquín Bravo Panadero

Directores: Dr. César I. García Osorio
José Francisco Díez Pastor

C. ESPECIFICACIÓN DE DISEÑO

C.1 Introducción

En este apartado se pretende dar una primera aproximación a la solución final del problema. Partiendo del análisis realizado en el apartado anterior, ahora se va a especificar cómo van a interactuar los objetos entre sí para dar la solución al problema, en contraposición a la anterior etapa de análisis dónde sólo se especificaba la funcionalidad. La frontera entre la finalización del análisis y el comienzo del diseño es difusa ya que el modelo va evolucionando y refinándose en cada paso.

En este apartado se marcará el camino a seguir hacia la solución final, y se tomarán decisiones muy importantes dentro de la arquitectura, los datos o la interfaz. Los detalles de diseño son muy importantes para conseguir unos factores de calidad externos e internos que permitan obtener un producto final de calidad.

Como se ha comentado con anterioridad, se va a realizar un diseño orientado a objetos. Esta metodología permite abordar el problema de una manera eficaz, y trabajar con clases y objetos que representan las abstracciones de los entes que se manifiestan en el dominio del problema, permitiéndonos así tener una comprensión mucho más clara de cómo funciona el sistema final, y de cómo están relacionadas sus piezas.

El diseño es una pieza clave para el correcto desarrollo de un proyecto software, ya que facilita llevar a cabo una estructuración modular, permitiendo identificar cada elemento del programa. Esto hace posible encontrar todos aquellos puntos peligrosos a los que el desarrollo podría enfrentarse en la fase de construcción.

En este apartado se definirá la estructura de paquetes de la aplicación y las relaciones que tienen entre ellos. Después, se desglosará cada paquete con los diagramas de clases, permitiendo ver y comprender el diseño con más detalle. Estos diagramas no incluirán todas las clases y atributos, sino que, por temas de legibilidad, sólo contendrán aquellas que sean más importantes para el funcionamiento del programa, ocultando información superflua.

C.2 Ámbito del software

El objetivo del sistema es la construcción de una aplicación que permita trabajar con imágenes radiográficas, pudiendo entrenar un clasificador tanto desde cero (es decir, con una serie de imágenes etiquetadas) como a partir de un fichero *ARFF*, así como analizar esas radiografías para determinar si existen defectos.

Se diseñará una interfaz gráfica que deberá disponer de las características que se definieron en la fase de análisis previa. Dicha interfaz deberá ser intuitiva y fácil de usar, aportando al usuario la máxima información posible para que pueda utilizarla eficientemente.

Las restricciones de diseño a aplicar sobre el desarrollo son:

- Arquitectura basada en objetos.
- Aplicación extensible.
- Interfaz sencilla e intuitiva.
- Máxima información de cara al usuario.

C.3 Diseño Arquitectónico

El diseño arquitectónico tiene como objetivo desarrollar la estructura modular que representa las relaciones entre los módulos, combinando la estructura del programa con la de los datos a través de interfaces que permiten el flujo de éstos.

La arquitectura de la aplicación depende del problema a tratar. La mayoría de las veces se utilizan varios estilos arquitectónicos para la solución.

C.3.1 Estilos arquitectónicos utilizados

Tras un estudio de distintos estilos arquitectónicos, hemos considerado que los más convenientes para nuestra aplicación son los que se muestran a continuación.

■ Arquitectura en capas

Este tipo de arquitectura distribuye jerárquicamente los roles y las responsabilidades. De esta forma, se proporciona una división de los problemas a resolver. Los roles indican cómo una capa se relaciona con otra, mientras que las responsabilidades indican su funcionalidad [2]. Entre sus características, podemos destacar:

- Descomposición de los servicios.
- Las capas pueden permanecer en una misma máquina o en equipos distintos.
- Los componentes de cada capa se comunican con otras capas mediante interfaces.
- Se diferencia claramente la funcionalidad de cada capa.
- Se trata de una abstracción a muy alto nivel, por lo que no se conocen tipos de datos, atributos, métodos e implementaciones.
- Las capas inferiores no tienen dependencias

Como principales ventajas, tenemos la abstracción, el aislamiento, el rendimiento y la testeabilidad, al ser cada capa independiente de las demás.

Esta arquitectura debería usarse cuando:

- Se tienen construidas capas de otra aplicación y pueden ser reutilizables.
- La aplicación es muy compleja y el diseño requiere la separación de funcionalidad.
- Se quiere implementar reglas de negocio complicadas.
- Se deben soportan diferentes tipos de clientes.

En (ver figura C.1) podemos ver cómo se estructura.

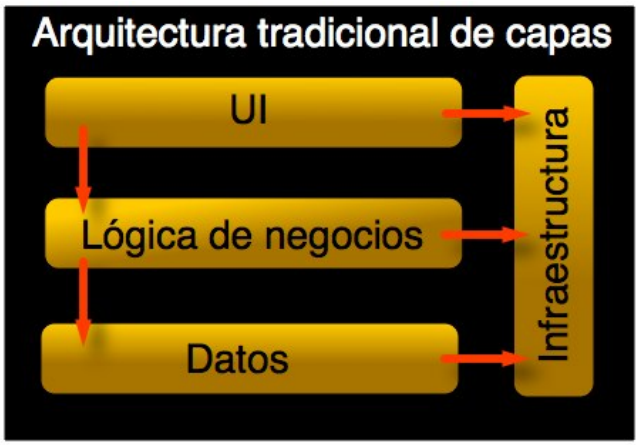

Figura C.1.: Arquitectura en capas [2]

C.3.2 Arquitectura final de la aplicación

Hemos estructurado nuestra aplicación siguiendo la arquitectura en capas, en la cual distinguimos:

- La capa de presentación o interfaz es la capa encargada de interactuar con el usuario. Permite, por ejemplo, que éste pueda visualizar los resultados de los análisis de imágenes y se encarga también de recibir los datos de entrada del usuario.
- La capa de lógica de negocio contiene todas las reglas del dominio de nuestra aplicación. Esta capa en ocasiones recibe otros nombres como dominio, modelo de negocio, modelo de dominio, etc. En nuestro caso, es la capa del modelo, que se encarga de manejar la lógica de la aplicación.
- La capa de acceso a datos se encarga de obtener los datos alojados en sistemas de almacenamiento persistente. En nuestro caso, abre las imágenes y se encarga de gestionar físicamente los archivos *ARFF*.

C.3.3 Patrones de diseño utilizados

Hemos realizado una clasificación de los patrones de diseño de acuerdo a la capa lógica a la que pertenecen: Acceso a Datos, Dominio o Presentación [15].

■ Patrones en la capa de Modelo del Dominio

Fachada

Hemos utilizado el patrón Fachada para separar la parte de interfaz de usuario de la capa de dominio. Esta fachada encapsula la lógica de la aplicación y va a ser la encargada de manejar los procesos de análisis y entrenamiento, controlando el comportamiento de las ventanas. En la imagen (ver figura C.2) se puede ver su estructura. Tiene las siguientes características:

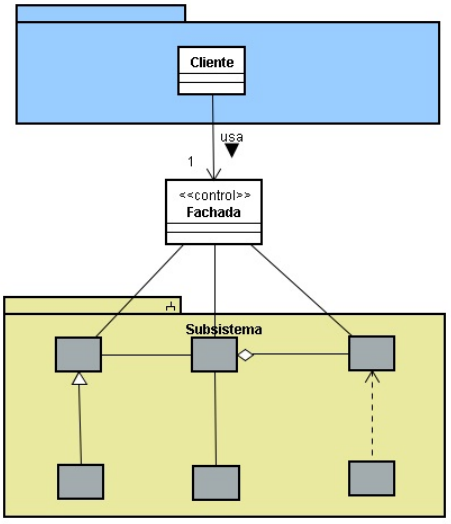

Figura C.2.: Patrón Fachada - Diagrama de clases [15]

- **Problema**
  - Existe mucha dependencia entre las clases que representan una abstracción y las clases clientes. Las dependencias añaden una complejidad notable a los clientes.
  - Se quiere simplificar las clases clientes.
  - Se quiere poner una barrera entre una clase cliente y un conjunto de clases y relaciones que implementan una abstracción.
- **Solución**
  - Se proporciona un objeto adicional reutilizable que oculta gran parte de la complejidad del trabajo de las clases. El cliente sólo tiene que relacionarse con este nuevo objeto.
  - Los clientes se comunican con el subsistema a través de la fachada, que reenvía las peticiones a los objetos del subsistema apropiados y puede realizar también algún trabajo de traducción.
  - Los clientes que usan la fachada no necesitan acceder directamente a los objetos del sistema.
- **Conclusiones**
  - Los clientes no necesitan conocer cómo se relacionan, ni cómo se crean las clases que proporcionan el servicio, solo se comunican a través del los objetos Fachada.
  - Reduce o elimina el acoplamiento entre las clases cliente y las clases que implementan la abstracción. Se podrían cambiar las clases que implementan la abstracción sin ningún impacto en las clases clientes. Ayuda a dividir un sistema en capas y reduce dependencias de compilación.

- Los clientes que necesiten acceder directamente a los objetos que implementan la abstracción, pueden acceder a ellos.

En nuestro caso, como ya hemos dicho, utilizamos una clase llamada Fachada que es la que separa las capas y la que encapsula la lógica (ver figura C.3)

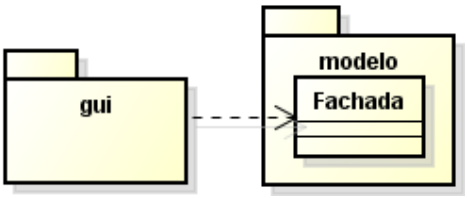

Figura C.3.: Diagrama de clases de nuestra Fachada

Singleton

Hemos utilizado el patrón Singleton para poder instanciar únicamente un objeto de tipo Fachada, que es el encargado de manejar la lógica de negocio. En (ver figura C.4) podemos ver cómo se estructura este patrón. Tiene las siguientes características:

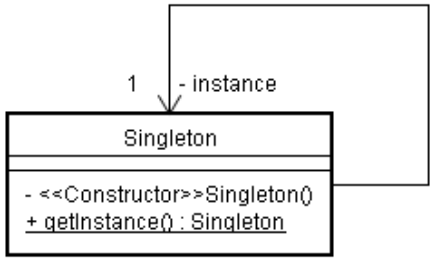

Figura C.4.: Patrón Singleton - Diagrama de clases [15]

- **Problema**

  Algunas clases deberían tener sólo una instancia. Estas clases están generalmente relacionadas con el manejo de un determinado recurso.

- **Solución**

  - Involucra una única clase.
  - La clase Singleton tiene una variable estática que referencia a la única instancia de la clase.
  - Para prevenir que los clientes creen más instancias de la clase se declara el constructor privado.

- La instancia de la clase se puede crear cuando la clase se carga.
- Para permitir el acceso a la instancia, la clase proporciona un método estático, típicamente llamado getInstance().

- **Conclusiones**
  - Sólo existe una instancia de la clase Singleton.
  - Otras clases que utilizar la instancia de la clase Singleton lo deben hacer invocando el método getInstance().

Estrategia

Hemos utilizado el patrón estrategia para encapsular los diferentes algoritmos que usa nuestra aplicación. Hemos necesitado tres estrategias: una para los tipos de ventana, otra para los algoritmos de extracción de características y otra para los algoritmos de preprocesamiento. Con esto buscamos que se puedan añadir fácilmente nuevos algoritmos.

En (ver figura C.5) podemos ver cómo se estructura este patrón. Tiene las siguientes características:

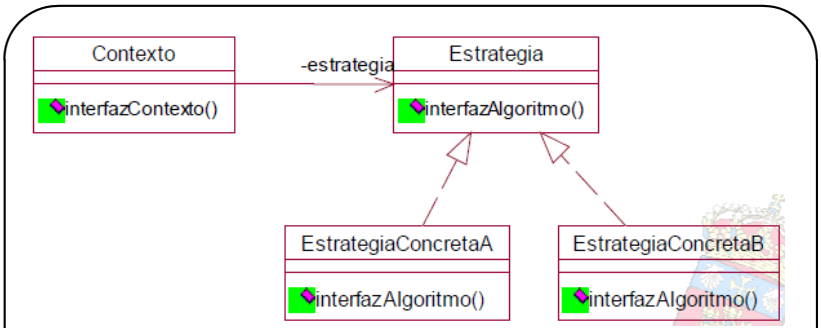

Figura C.5.: Patrón Estrategia - Diagrama de clases [15]

- **Problema**
  - Muchas clases relacionadas difieren solo en su comportamiento.
  - Se necesitan distintas variantes de un algoritmo.
  - Un algoritmo usa datos que el cliente no debería conocer. Se quiere evitar exponer estructuras complejas y dependientes del algoritmo.
  - Una clase define múltiples comportamientos y estos se representan con múltiples sentencias condicionales en sus operaciones.

- **Solución**
  - Define una familia de algoritmos, encapsula cada uno de ellos y los hace intercambiables. Permite que un algoritmo varíe independientemente de los clientes que lo usan.
  - Se declara una interfaz común para todos los algoritmos soportados

- Los clientes que usan la fachada no necesitan acceder directamente a los objetos del sistema.

Como ya hemos dicho, nosotros tenemos tres estrategias, por lo que necesitamos tres superclases que contendrán las operaciones comunes de cada estrategia. De ellas heredarán una serie de clases, que son las que implementan los algoritmos concretos, como puede ser el desplazamiento de una ventana deslizante o la implementación de las características de Haralick. En (ver figura C.6) se puede ver una simplificación de nuestras clases.

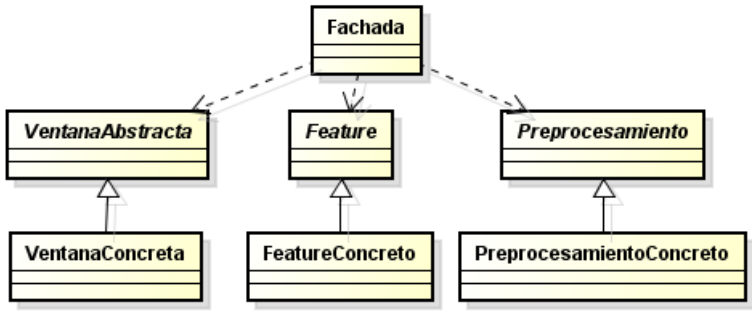

Figura C.6.: Diagrama de clases de nuestras Estrategias

C.3.4 Diagrama de clases de diseño

FALTA

C.4 Diseño de la interfaz

En este apartado, se cierra la parte de diseño de la interfaz gráfica, y se resaltan los aspectos más relevantes de este proceso.

C.4.1 Ventana principal

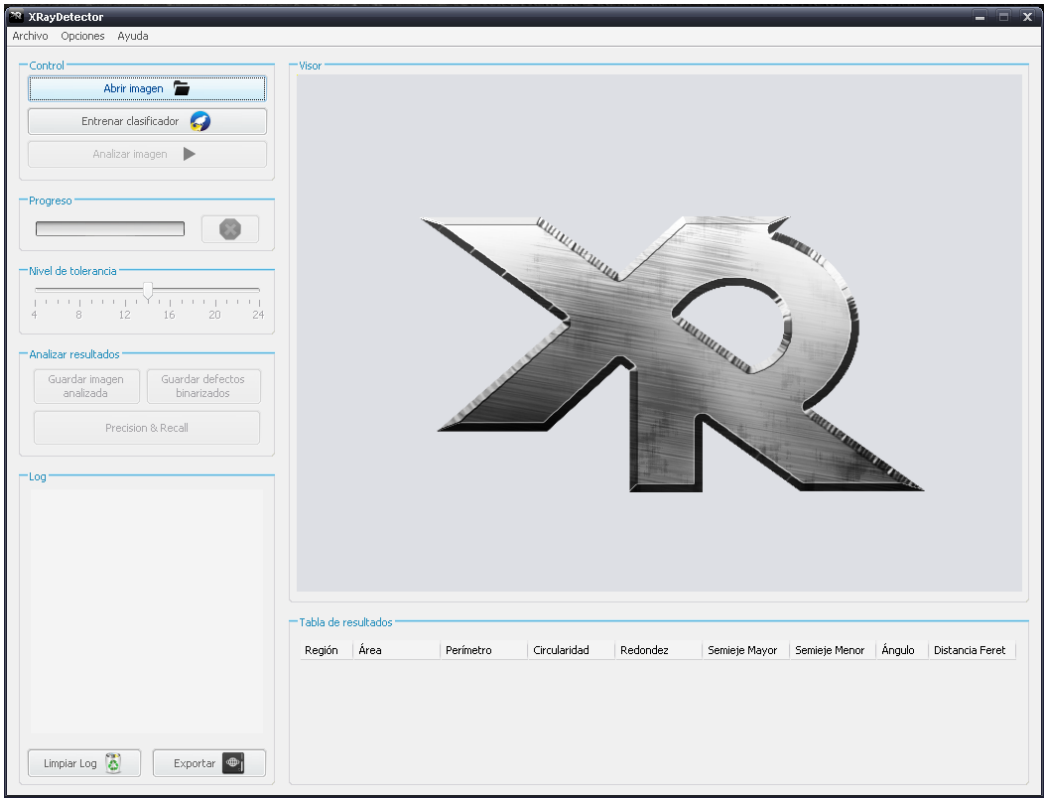

Figura C.7.: Diseño de la ventana principal

Desde la ventana principal podemos acceder a todos los casos de uso mediante botones y menú. Se ha preferido esta distribución a, por ejemplo, distintas ventanas, para tenerlo todo en un mismo lugar. Para evitar confusiones, se ha elegido una estrategia de habilitación/deshabilitación de los botones según se puedan hacer o no determinadas acciones.

C.4.2 **Ventana de opciones avanzadas**

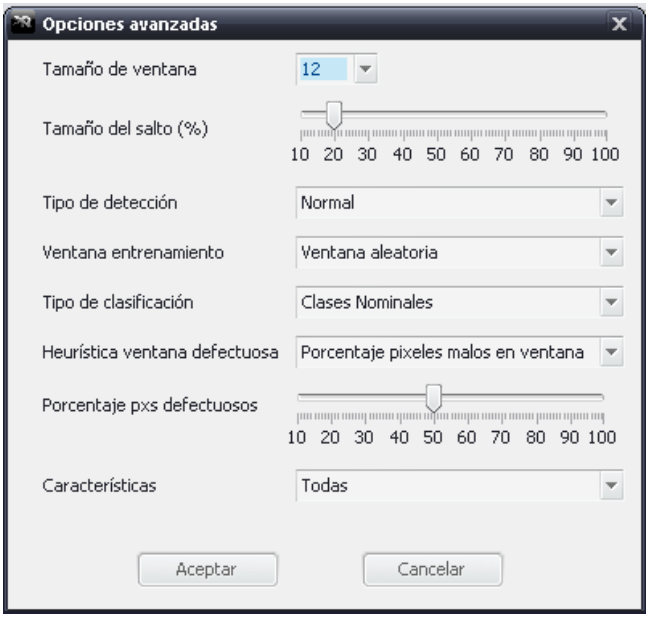

Figura C.8.: Diseño de la ventana de opciones avanzadas

Desde esta ventana se permite cambiar todas las opciones del programa. Se accede a ella mediante el menú de opciones de la ventana principal.

C.4.3 **Ventana de Precision & Recall**

Figura C.9.: Diseño de la ventana de precision & recall

En esta ventana se muestran los resultados de precision & recall. Se accede a ella mediante el botón de la ventana principal, que sólo se puede pulsar cuando ha acabado un análisis.

C.5 Diseño procedimental

En el presente apartado, se recogerán algunos de los algoritmos más importantes que hemos utilizado, eliminando cualquier ambigüedad.

Nos hemos basado en los casos de uso que se explicaron en el anexo II de esta memoria. A continuación, se explica cada uno por separado, usando para ello diagramas de secuencia que muestran la interacción entre los diferentes objetos.

FALTA

C.6 Referencia cruzada a los requisitos

De cara a la trazabilidad del sistema, es interesante incluir matrices que relacionen los requisitos funcionales con los elementos de diseño. De esta forma puede observarse rápidamente si se han satisfecho todos los requisitos de la aplicación.

FALTA

C.7 Pruebas

En este apartado se recogen las pruebas de integración del sistema y aspectos relevantes del diseño de las mismas.

Las pruebas de integración testean los módulos de la aplicación de forma combinada, es decir, como un grupo. Estas pruebas se realizan testeando un conjunto de elementos unitarios.

C.7.1 Pruebas de integración

FALTA

C.8 Entorno tecnológico del sistema

Este entorno establece el equipo físico, el equipo lógico y las comunicaciones del sistema software. Los detalles de la parte física se indican en el siguiente apartado.

Necesitaremos tener instalada la máquina virtual de Java. La aplicación es monolítica, por lo que sólo estará desplegada en un único equipo.

Es recomendable tener instalado también un navegador web para poder ver los informes que se exporten.

Las dependencias entre la aplicación y las librerías que utiliza se muestran en (ver figura C.10)

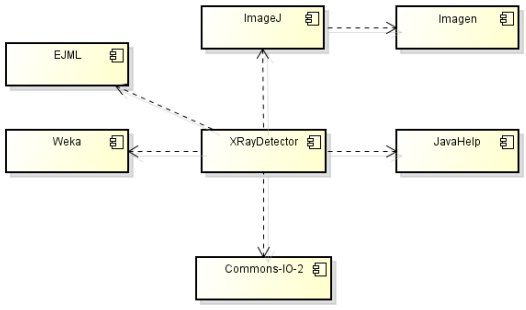

Figura C.10.: Diagrama de componentes

C.9 Plan de desarrollo e implantación

La aplicación se implantará en una misma máquina que proporcione todas las funcionalidades requeridas, ya que, como hemos dicho, es monolítica.

En la imagen (ver figura C.11) se muestra el diagrama de despliegue que representa la implantación de la aplicación en la máquina cliente.

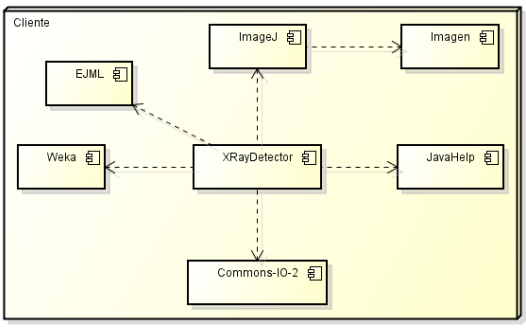

Figura C.11.: Diagrama de despliegue

Universidad de Burgos

Escuela Politécnica Superior

Ingeniería Informática

Área de Lenguajes y Sistemas Informáticos

Anexo IV - Manual del programador

X-RayDetector: Detección automática de defectos en piezas metálicas mediante análisis de radiografías

Adrián González Duarte y Joaquín Bravo Panadero

Directores: Dr. César I. García Osorio
José Francisco Díez Pastor

D. MANUAL DEL PROGRAMADOR

D.1 Introducción

Este anexo explica en detalle la fase de implementación del proyecto.

Se realiza la documentación de las librerías utilizadas y las creadas específicamente para la aplicación, un manual del programador para que otras personas puedan trabajar en el proyecto en un futuro y una descripción de las pruebas unitarias del sistema.

D.2 Documentación de las bibliotecas

D.2.1 Bibliotecas de Java

Java es un lenguaje de programación orientado a objetos, desarrollado por Sun Microsystems a principios de los años 90.

El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria.

En concreto, nosotros hemos utilizado *Java 7*, por lo que es necesario descargar el *Development Kit* correspondiente a esta versión.

D.2.2 ImageJ

ImageJ permite ser usado programáticamente mediante llamadas a un JAR. Esto nos ha permitido trabajar con las imágenes de forma sencilla, por ejemplo, estableciendo regiones de interés en una imagen y extrayendo información de sus píxeles.

En concreto, hemos usado la versión 1.47n, que se corresponde con el archivo *ij.jar*.

D.2.3 Weka

Weka[48] es una plataforma de software para aprendizaje automático y minería de datos, diseñada por la Universidad de Waikato. Está escrita en *Java*.

Nosotros hemos utilizado *Weka* usando su JAR, lo que permite realizar llamadas a sus métodos dentro de nuestro código, como para, por ejemplo, construir y usar clasificadores. Esto permite realizar estas tareas de una forma muy sencilla, sin necesidad de programar estos métodos manualmente.

D.2.4 JavaHelp

JavaHelp es una expansión de Java que facilita la programación de las ventanas de ayuda en las aplicaciones java.

Con JavaHelp se pueden crear las ventanas típicas de ayuda de las aplicaciones informáticas, en las que sale en el lado izquierdo un panel con varias pestañas: índice de contenidos, búsqueda, temas favoritos, índice alfabético, etc., mientras que en el lado derecho sale el texto de la ayuda.

Las ventanas de ayuda pueden lanzarse directamente con la pulsación de botones en la aplicación, o bien por medio de la pulsación de la tecla F1, mostrando la ayuda correspondiente a la ventana sobre la que estamos trabajando.

Las ventanas de ayuda de JavaHelp se configuran por medio de varios ficheros en formato XML. Los textos de ayuda que se quieran mostrar se escribirán en ficheros con formato HTML.

JavaHelp no se incluye en la JDK, ni en la JRE, sino que debe conseguirse como un paquete externo.

La biblioteca correspondiente al módulo de JavaHelp se corresponde con el archivo jh.jar.

D.2.5 Apache Commons IO

Esta librería permite realizar algunas operaciones con ficheros, como es la fusión de uno o más ficheros de texto, o la exportación de un cierto texto a un fichero externo de una forma muy sencilla, evitándonos problemas.

Hemos usado la versión 2.4, y se corresponde con el archivo *commons-io-2.4*.

D.2.6 EJML

Esta librería ha sido usada sólo para el cálculo de una de las caracteristicas de Haralick que requería previamente el cálculo de unos autovalores. Estos cálculos son computacionalmente muy exigentes, y la solución que dieron los desarrolladores anteriores no era buena. Esta librería realiza cálculos con matrices de forma muy eficiente, pero pese a ello, los cálculos que necesitamos ralentizan la ejecución de nuestra aplicación.

Hemos usado la versión 0.21, que se corresponde con el archivo *ejml-0.21*.

D.3 Código fuente

El código fuente de la aplicación XRayDetector se puede encontrar en la carpeta XRayDetector/src.

Dentro de la carpeta src, los archivos pertenecientes al código fuente se encuentran organizados en carpetas correspondientes a los diferentes paquetes que conforman la estructura de ficheros Java de la aplicación.

Esta estructura está organizada de la siquiente forma:

FALTA

D.3.1 Recursos necesarios por el código fuente

La carpeta *res* contiene los archivos para la correcta ejecución y funcionamiento de la aplicación. Se encuentra estructurada en subcarpetas, que son:

- Arff: contiene ficheros *ARFF* de ejemplo.
- Ayuda: Se corresponde con el módulo de ayuda en línea que utiliza la aplicación mediante *JavaHelp*.
- Config: contiene un único fichero (*config.properties*) con las opciones elegidas por el usuario.
- Img: contiene, por un lado, imágenes de ejemplo para usar la aplicación, y, por otro lado, los iconos necesarios para la interfaz de la aplicación (en la carpeta *app*).
- Log: contiene el archivo generado por el gestor de log integrado en la aplicación. Además, cuando el usuario exporta el contenido del log de la interfaz, se guarda automáticamente en la carpeta *html* dentro de esta misma carpeta.
- Model: contiene clasificadores de ejemplo para poder usar la aplicación.

La carpeta *lib* contiene las bibliotecas necesarias para la compilación y ejecución de la aplicación. Estas librerías se corresponden con los archivos .JAR, descritos en el apartado D.2.

El código fuente de las pruebas se encuentra en la carpeta XRayDetector/test.

La documentación interna de las clases tras ser generada es almacenada en el directorio XRayDetector/docs/javadoc.

La documentación correspondiente al docheck referente a la documentación interna de las clases tras ser generada es almacenada en el directorio XRayDetector/docs/docCheck.

D.4 Manual del programador

Este manual pretende ser una guía de referencia para futuros programadores de la aplicación, facilitándoles en la medida de lo posible, la creación de nuevos componentes.

D.4.1 Agregar nuevos elementos a la aplicación

A continuación se va a mostrar un ejemplo para añadir nuevos componentes a la aplicación, creando nuevos tipos de ventana, nuevos algoritmos de extracción de características y nuevos algoritmos de preprocesamiento.

■ Creación de nuevos elementos en las estrategias

Como ya hemos visto, para las ventanas, extracción de características y preprocesamiento se han usado sendos patrones estrategia. Por lo tanto, el primer paso es simplemente incluir la nueva clase en esta estructura, heredando de la superclase que corresponda. Después, hay que implementar

los métodos abstractos de estas superclases, que son realmente donde va a estar la funcionalidad específica.

En la extracción de características, es necesario guardar un vector con los nombres de cada descriptor en particular (*headVector*) y otro, del mismo tamaño, con los valores de cada descriptor, en formato *double*.

Para integrar una nueva ventana en el funcionamiento de la aplicación, es necesario añadir una opción en las opciones avanzadas y, después, los métodos que llaman a las ventanas (ejecutar entrenamiento, ejecutar análisis) deberán controlarlas.

Para integrar un nuevo algoritmo de extracción de características, habría que llamar a sus métodos dentro e la clase VentanaAbstracta, que es la que contiene los cálculos. Además, se hace necesario añadir los nombres de los descriptores en la cabecera de los *ARFF*.

Para integrar un nuevo algoritmo de preprocesamiento, sería necesario crear una nueva opción, ya que de momento sólo hay uno y no se ha implementado esta opción, aunque la estructura está pensaba para albergar nuevos algoritmos.

D.4.2 Modificación del módulo de ayuda en línea de la aplicación y del motor de búsqueda

Para realizar el módulo de ayuda online de la aplicación, hemos utilizado la biblioteca JavaHelp, la cual ya hemos reseñado en este mismo anexo de la memoria.

Para poder usar JavaHelp, necesitaremos los siguientes ficheros:

- Ficheros html: en estos ficheros escribiremos la ayuda de la aplicación. Se usa una codificación html estándar, y podemos poner la información de ayuda correspondiente a las ventanas de la aplicación.
- Mapa de JavaHelp: este fichero contiene los nombres de los html junto con la clave que le damos a cada uno de ellos.

  Tendremos un mapID por cada html que queremos que se muestre. En nuestro caso, es el fichero «mapa.jhm» contenido en el directorio XRayDetector/res/ayuda.
- Tabla de contenidos: en él incluiremos los capítulos y subcapítulos de los que consta nuestra ayuda. En nuestro caso, se trata del fichero «toc.xml» que se encuentra en el directorio de ayuda.
- Fichero HelpSet: es el fichero principal de la ayuda. En él se indica qué se mostrará en la ayuda. En nuestro caso, sólo queremos que se muestren la tabla de contenidos y un motor de búsqueda.

  Hay que indicar, además, cuál es el archivo de mapas. Es el fichero «ayuda.hs» del directorio de ayuda.
- Motor de búsqueda: para realizar el motor de búsqueda, debemos tener descomprimida la biblioteca de JavaHelp.

  Además de meter la correspondiente parte del motor en el fichero del HelpSet, debemos introducir el siguiente comando mediante línea de comandos, estando en el directorio de la ayuda:

```
java -jar path_java_help/jh2.0/javahelp/bin/jhindexer.jar
```

Donde el *path_java_help* es el directorio en el que hemos descomprimido la biblioteca de JavaHelp. Una vez realizado, se habrá creado una carpeta llamada *JavaHelpSearch*, que es la que contiene el motor de búsqueda.

D.5 Pruebas unitarias

FALTA

Universidad de Burgos

Escuela Politécnica Superior

Ingeniería Informática

Área de Lenguajes y Sistemas Informáticos

Anexo V - Manual del usuario

X-RayDetector: Detección automática de defectos en piezas metálicas mediante análisis de radiografías

Adrián González Duarte y Joaquín Bravo Panadero

Directores: Dr. César I. García Osorio
José Francisco Díez Pastor

E. MANUAL DEL USUARIO

E.1 Documento de instalación y configuración

E.1.1 Requisitos

El único requisito necesario para la ejecución de XRayDetector es tener instalada la máquina virtual de Java en el equipo.

E.1.2 Instalación y ejecución de la aplicación

Para realizar la instalación de X-Ray Detector, basta con copiar la carpeta de «XRayDetector» en el directorio deseado.

Para ejecutar la aplicación haz doble clic en el archivo «ejecutar.bat».

E.2 Manual de usuario

E.2.1 Ventana principal de la aplicación

Al iniciar la aplicación aparecerá la ventana de (ver figura E.1), que se corresponde con la ventana principal de XRayDetector.

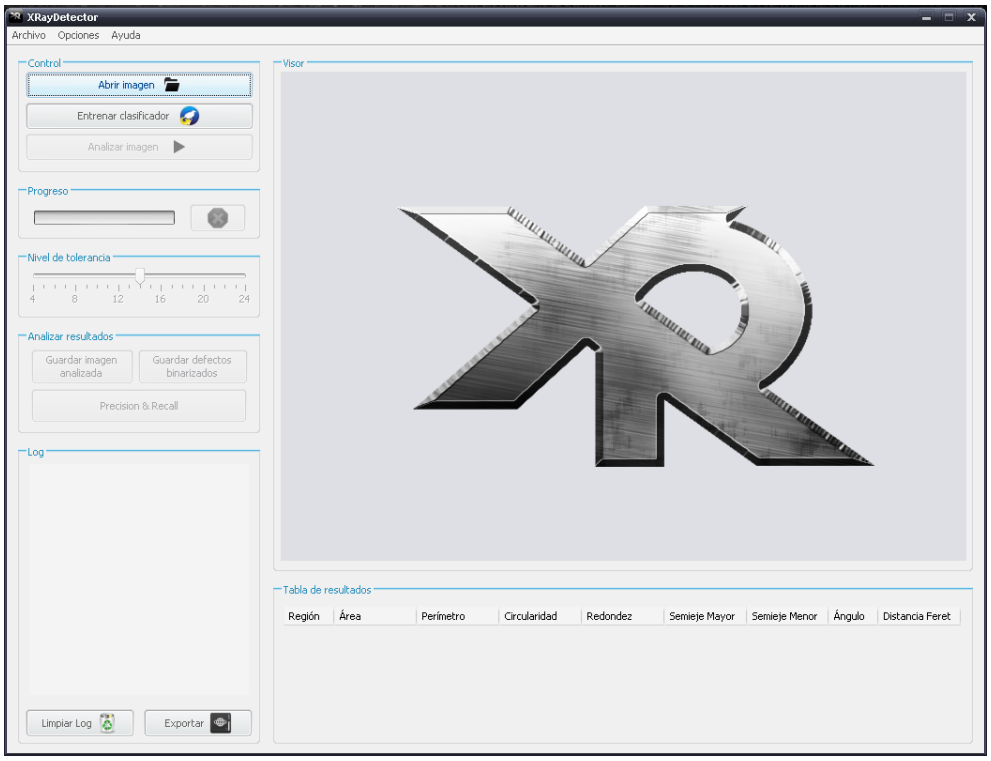

Figura E.1.: Ventana principal de XRayDetector

A continuación, se procede a describir los elementos que componen la ventana principal, que se encuentra dividida en múltiples secciones.

■ Control

Aquí nos podemos encontrar los controles principales de la aplicación.

- **Abrir imagen:** Permite cargar una imagen en la aplicación.
- **Entrenar clasificador:** Entrenar un clasificador a partir de un conjunto de imágenes o de un archivo ARFF.
- **Analizar imagen:** Analiza la imagen en busca de defectos.

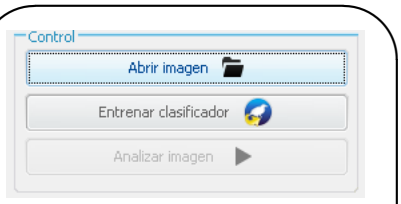

Figura E.2.: Menú control

Progreso

Indica el progreso que lleva la tarea que se está realizando la aplicación hasta que se complete.

- **Barra de progreso:** Indica cuál es el progreso actual de la tarea que se está realizando.
- **Botón Stop:** Detiene la tarea que se está realizando actualmente.

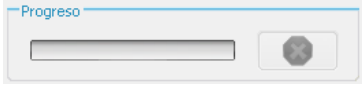

Figura E.3.: Menú progreso

Nivel de tolerancia

Indica el nivel de tolerancia que se utilizará para ajustar los bordes al defecto detectado, a mayor valor mayor precisión. Este valor puede ajustarse desplazando la barra de desplazamiento a izquierda y derecha.

Figura E.4.: Nivel de tolerancia

Analizar resultados

Permite analizar y exportar los resultados obtenidos tras el proceso de detección.

- **Guardar imagen analizada:** Permite guardar una copia de la imagen mostrada en el visor al disco duro.
- **Guardar imagen binarizada:** Guarda una copia del defecto detectado sobre la imagen a través de una imagen binarizada en el disco duro.
- **Calcular precision and recall:** Calcula los valores de precision and recall de la imagen sobre los resultados obtenidos.

Figura E.5.: Analizar resultados

■ Log

Muestra al usuario información referente al estado de la aplicación y los resultados tras ejecutar diversas operaciones en XRayDetector.

- **Limpiar log:** Borra el contenido actual del log.
- **Exportar log:** Exporta el contenido actual del log a un archivo HTML.

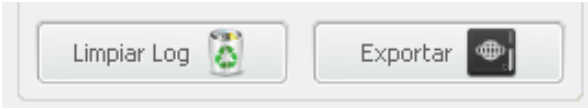

Figura E.6.: Control de Log

■ Visor

Aquí se muestra la imagen cargada en la aplicación, así como el resultado del proceso de detección de defectos. Una vez analizada la imagen, se marcarán los defectos en la imagen y podrán ser seleccionados a partir directamente sobre la misma o sobre la tabla de resultados con la lista de defectos y sus características.

Figura E.7.: Visor

■ Tabla resultados

Aquí se muestra una lista de los defectos detectados durante el proceso de detección, así como una colección de características geométricas asociadas al defecto. Si se selecciona una fila en la tabla de resultados, el defecto se coloreará en el visor indicando con qué defecto se corresponde. De la misma forma, si se selecciona un defecto sobre el visor mediante la combinación de la tecla *control* y click del ratón, se resaltará la fila de la tabla a la que corresponde.

■ Barra de menús

Aquí se encuentran algunas opciones adicionales que pueden realizarse sobre la aplicación.

- **Menú Archivo:** Permite salir de la aplicación mediante la opción Salir.
- **Menú Opciones:** Permite acceder a la configuración de opciones avanzadas de la aplicación.
- **Menú Ayuda:** Permite acceder a la ayuda en línea de la aplicación, así como información relativa a la misma.

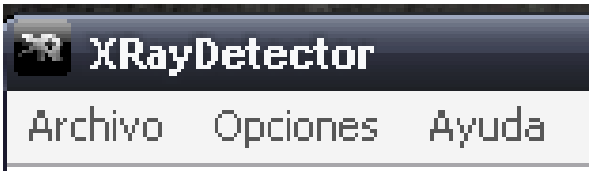

Figura E.8.: Barra de menús

En los siguientes apartados, iremos explicando en detalle el funcionamiento completo de la aplicación, a partir de los principales componentes ya vistos.

E.2.2 Configuración de la aplicación

Mediante el menú **Opciones**, de la barra de menú y seleccionando **Opciones Avanzadas**, podemos acceder a la configuración de las opciones avanzadas de la aplicación que nos permitirá cambiar diversos parámetros sobre la misma. Vamos a explicar qué significa cada opción.

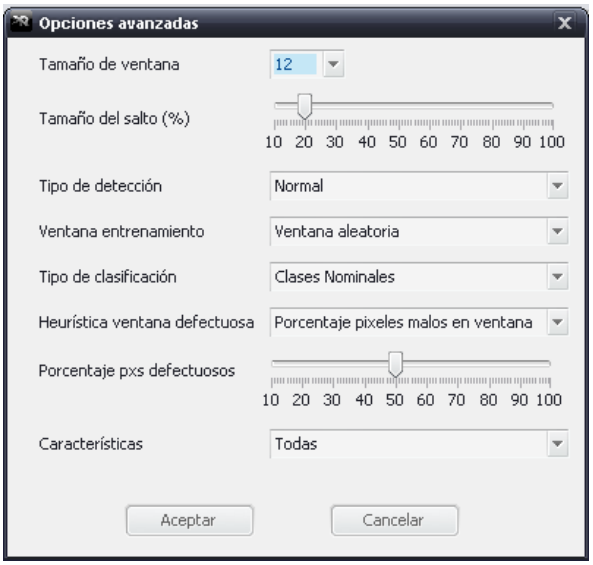

Figura E.9.: Opciones avanzadas

- **Tamaño de la ventana:** Permite especificar el tamaño de ventana que se utilizará durante los procesos de entrenamiento y detección de defectos. Los tamaños que la aplicación permite seleccionar son: 12×12, 16×16, 24×24 y 32×32.
- **Salto de la ventana:** Permite especificar en un porcentaje cuanto avanzará la ventana respecto a su tamaño salto a salto. Este valor puede modificarse mediante la barra de desplazamiento que puede ajustarse entre unos valores comprendidos entre el 10 % y el 100 % del tamaño de la ventana.
- **Tipo de detección:** Especifica el tipo de detección que se llevará a cabo durante el proceso de análisis:

  1. Normal: La detección se realiza sin tener en cuenta los falsos positivos. El resultado de la detección no se filtra por lo que se devuelven todas las posibles ventanas marcadas como defecto.
  2. Normal + Umbrales locales: La detección se realiza igual que en el modo normal, pero luego se filtran aquellos píxeles marcados como defecto haciendo una intersección de los mismos con el resultado del filtro de umbrales locales. Aquellos píxeles marcados como defectuosos que también están marcados como defectuosos en el filtro de umbrales locales son los que se mantienen.
  3. Blancos en umbrales locales: Primero se calculan los umbrales locales de la imagen y se saca una lista de píxeles en las regiones candidatas a albergar defectos. Durante el proceso de detección, se van considerando píxeles de cada región teniendo en cuenta

el tamaño de la misma, centrando en los que sí se consideren una ventana y calculando sus características.

- **Ventana de entrenamiento:** Especifica el tipo de ventana utilizada durante el proceso de entrenamiento:

    1. Deslizante: La ventana va recorriendo la imagen de forma secuencial sacando las características para crear el clasificador. El salto de la ventana viene determinado por el salto de la ventana indicado por el usuario.

    2. Aleatoria: La ventana va cogiendo muestras de forma aleatoria de la imagen para extraer sus características y construir el clasificador a partir de los datos obtenidos.

- **Tipo de clasificación:** Especifica el tipo de clasificación que se va a establecer cuando una ventana se ha analizado:

    1. Clases Nominales: Las ventanas se etiquetan indicando si son defectuosas o no mediante TRUE o FALSE.

    2. Regresión: Las ventanas se etiquetan indicando el numero de píxeles defectuosos que se han encontrado.

- **Heurística ventana defectuosa:** Las ventanas se marcarán como defectuosas en función al tipo de heurística seleccionada.

    1. Porcentaje píxeles malos en la ventana: Si el porcentaje de píxeles defectuosos detectados en la ventana es superior a un tanto por ciento del total de píxeles de la ventana, siendo este porcentaje definido por el usuario mediante la barra de desplazamiento de porcentaje de píxeles defectuosos, la ventana es marcada como defectuosa.

    2. Porcentaje de vecinos defectuosos respecto al píxel central: Si el porcentaje de píxeles defectuosos detectados respecto a los vecinos del píxel central de la ventana es superior a un tanto por ciento del total de vecinos del píxel central, siendo este porcentaje definido por el usuario mediante la barra de desplazamiento de porcentaje de píxeles defectuosos, la ventana es marcada como defectuosa.

- **Porcentaje pixeles defectuosos:** Indica el porcentaje que se tomará como referencia en la heurística seleccionada por el usuario para determinar el umbral por el cual una ventana será clasificada como defectuosa o no.

- **Características:** Determina que características se seleccionarán para construir el clasificador:

    1. Todas: Se utilizan todas las características extraídas para construir el clasificador.

    2. Las mejores: Se utilizan aquellas características que se consideran mejores para construir el clasificador, ya que pueden existir características que no aporten información relevante al clasificador para discriminar los datos.

Universidad de Burgos

Escuela Politécnica Superior

Ingeniería Informática

Área de Lenguajes y Sistemas Informáticos

Apéndice A - Guía rápida de Version One

X-RayDetector: Detección automática de defectos en piezas metálicas mediante análisis de radiografías

Adrián González Duarte y Joaquín Bravo Panadero

Directores: Dr. César I. García Osorio
José Francisco Díez Pastor

Universidad de Burgos

Escuela Politécnica Superior

Ingeniería Informática

Área de Lenguajes y Sistemas Informáticos

Apéndice B - Licencia GNU GPL

X-RayDetector: Detección automática de defectos en piezas metálicas mediante análisis de radiografías

Adrián González Duarte y Joaquín Bravo Panadero

Directores: Dr. César I. García Osorio
José Francisco Díez Pastor

BIBLIOGRAFÍA

[1] Analyze menu. http://rsbweb.nih.gov/ij/docs/menus/analyze.html. URL: http://rsbweb.nih.gov/ij/docs/menus/analyze.html.

[2] Arquitectura de software. http://magmacode.net/arquitecturadesoftware.php. URL: http://magmacode.net/arquitecturadesoftware.php.

[3] Auto local threshold - fiji. http://fiji.sc/wiki/index.php/Auto_Local_Threshold. URL: http://fiji.sc/wiki/index.php/Auto_Local_Threshold.

[4] ImageJ user guide - IJ 1.46r | analyze menu. http://rsbweb.nih.gov/ij/docs/guide/146-30.html. URL: http://rsbweb.nih.gov/ij/docs/guide/146-30.html.

[5] Design Patterns: Elements of Reusable Object-Oriented Software. 1995.

[6] Extracción de caracteristicas de textura basada en la transformada Wavelet discreta. PhD thesis, Escuela superior de ingenieros de la universidad de Sevilla, 2009.

[7] M. D. Abramoff, P. J. Magelhaes, and S. J. Ram. Image processing with ImageJ. Biophotonics int, 11(7):36–42, 2004.

[8] Control Chaos [online]. URL: http://www.controlchaos.com.

[9] Etapas de Scrum [online]. URL: http://orione.wordpress.com/2008/02/24/agile-scrum-meet-with-sanjeev-augustine/.

[10] Proceso Scrum [online]. URL: http://www.1dot0.com/dev_process.html.

[11] ¿Qué es Scrum? [online]. URL: http://www.proyectosagiles.org/que-es-scrum/.

[12] Scrum [online]. URL: http://es.wikipedia.org/wiki/Scrum/.

[13] Rs Anand, P. Kumar, et al. Flaw detection in radiographic weldment images using morphological watershed segmentation technique. Ndt & e International, 42(1):2–8, 2009.

[14] Y. Bazi, L. Bruzzone, and F. Melgani. Image thresholding based on the em algorithm and the generalized gaussian distribution. Pattern recognition, 40(2):619–634, 2007.

[15] Carlos López Nozal. Apuntes de diseño y mantenimiento del software II, 2012.

[16] j.g. Daugman. Complete discrete 2-d gabor transforms by neural networks for image analysis and compression. Acoustics, speech and signal processing, ieee transactions on, 36(7):1169–1179, 1988.

[17] E.J. de Ramón Balmaseda. Transformaciones basadas en el algoritmo local binary pattern de imágenes capturadas con la kinect para clasificación facial. 2011.

[18] W. Dunham. Euler: The master of us all (dolciani mathematical expositions 22). Math. Assoc. Amer., Washington, 1999.

[19] R.M. Haralick, K. Shanmugam, and I.H. Dinstein. Textural features for image classification. Systems, Man and Cybernetics, IEEE Transactions on, 3(6):610–621, 1973.

[20] R.M. Haralick and L.G. Shapiro. Computer and Robot Vision, volume 1. Addison-Wesley, 1992.

[21] Ian H. Witten and Eibe Frank. Data Mining - Practical Machine Learning Tools and Techniques. MORGAN KAUFMANN PUBLISHERS, second edition edition, 2005.

[22] Juan José Rodríguez Díez. Apuntes de la asignatura minería de datos, 2012.

[23] A. Kandel. Introduction to pattern recognition: statistical, structural, neural, and fuzzy logic approaches, volume 32. World scientific Pub Co Inc, 1999.

[24] C. Koch and S. Ullman. Shifts in selective visual attention: towards the underlying neural circuitry. Hum Neurobiol, 4(4):219–27, 1985.

[25] L. Lamport. Latex: a document preparation system. 1989.

[26] LaTeX [online]. URL: http://en.wikipedia.org/wiki/LaTeX/.

[27] CervanTEX [online]. URL: http://goliat.mecanica.upm.es/cervantex/.

[28] Una Introducción no-tan-corta a LaTeX [online]. URL: http://tug.ctan.org/tex-archive/info/lshort/spanish/.

[29] LaTeX Wikibook [online]. URL: http://en.wikibooks.org/wiki/LaTeX.

[30] WYSIWYG: What You See Is What You Get [online]. URL: http://en.wikipedia.org/wiki/WYSIWYG.

[31] Licencia EPL [online]. URL: http://www.eclipse.org/legal/epl-v10.html.

[32] Álvar Arnáiz González. Biblioteca de algoritmos de selección de instancias y aplicación orientada a su docencia. PhD thesis, Universidad de Burgos, 2010. Tutores: César García Osorio and Juan José Rodriguez.

[33] S. Belaifa M. Tridi and N. Nacereddine. Weld defect classification using em algorithm for gaussian mixture model. 2005.

[34] Marcos Martin. Tecnicas clasicas de segmentacion de la imagen. 2004.

[35] D. Mery. Automated detection of welding defects without segmentation. Materials Evaluation, 5:657–663, 2011.

[36] D. Mery, I. Lillo, H. Loebel, V. Riffo, A. Soto, A. Cipriano, and J.M. Aguilera. Automated fish bone detection using x-ray imaging. Journal of Food Engineering, 2011.

[37] S. Montabone and A. Soto. Human detection using a mobile platform and novel features derived from a visual saliency mechanism. Image and Vision Computing, 28(3):391–402, 2010.

[38] E. Niebur. Saliency map. Scholarpedia, 2(8):2675, 2007.

[39] N. Otsu. A threshold selection method from gray-level histograms. Ieee transactions on systems man and cybernetics, 9(1):62–66, 1979.

[40] K. Pearson. On lines and planes of closest fit to systems of points in space. Philosophical magazine, 2:559–572, 1901.

[41] M. Presutti. La matriz de co-ocurrencia en la clasificación multiespectral: tutorial para la enseñanza de medidas texturales en cursos de grado universitario. 4ª jornada de educação em sensoriamento remoto no âmbito do mercosul, são leopoldo, brasil, 2004.

[42] J. Sauvola and M. Pietikäinen. Adaptive document image binarization. PATTERN RECOGNITION, 33:225–236, 2000.

[43] Mehmet Sezgin and Bulent Sankur. Survey over image thresholding techniques and quantitative performance evaluation. Journal of Electronic Imaging, 13(1):146–168, 2004. URL: +http://dx.doi.org/10.1117/1.1631315.

[44] M. Pietikainen T. Ojala and T. Maenpaa. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. Ieee trans. on pattern analysis and machine intelligence, 24(7):971–987, 2002.

[45] Ioannis Valavanis and Dimitrios Kosmopoulos. Multiclass defect detection and classification in weld radiographic images using geometric and texture features. Expert Systems with Applications, 37(12):7606 – 7614, 2010. URL: http://www.sciencedirect.com/science/article/pii/S0957417410003829, doi:10.1016/j.eswa.2010.04.082.

[46] R. Vilar, J. Zapata, and R. Ruiz. An automatic system of classification of weld defects in radiographic images. Ndt & e International, 42(5):467–476, 2009.

[47] X. Wang, B.S. Wong, and C.S. Tan. Recognition of welding defects in radiographic images by using support vector machine classifier. A research journal of applied sciences, engineering and technology, 2(3), 2010.

[48] Weka (Waikato Environment for Knowledge Analysis) [online]. URL: http://www.cs.waikato.ac.nz/ml/weka/.

[49] N. Wiener. Extrapolation, interpolation, and smoothing of stationary time series: with engineering applications. Journal of the American Statistical Association, 47(258), 1949.

[50] Wikipedia. Clasificadores (matematico) — wikipedia, la enciclopedia libre, 2011. URL: http://es.wikipedia.org/w/index.php?title=Clasificadores_(matem%C3%A1tico)&oldid=43291813.

[51] Wikipedia. Local binary patterns — wikipedia, the free encyclopedia, 2011. URL: http://en.wikipedia.org/w/index.php?title=Local_binary_patterns&oldid=459646939.

[52] Wikipedia. Ensayo no destructivo — wikipedia, la enciclopedia libre, 2013. URL: http://es.wikipedia.org/w/index.php?title=Ensayo_no_destructivo&oldid=55705663.

[53] Wikipedia. Image segmentation — wikipedia, the free encyclopedia, 2013. URL: http://en.wikipedia.org/w/index.php?title=Image_segmentation&oldid=549435126.

[54] Wikipedia. Precision and recall — wikipedia, the free encyclopedia, 2013. URL: http://en.wikipedia.org/w/index.php?title=Precision_and_

recall&oldid=555149430.

[55] Wikipedia. Radiografía — wikipedia, la enciclopedia libre, 2013. URL: http://es.wikipedia.org/w/index.php?title=Radiograf%C3%ADa&oldid=56282751.

[56] Wikipedia. Thresholding (image processing) — wikipedia, the free encyclopedia, 2013. URL: http://en.wikipedia.org/w/index.php?title=Thresholding_(image_processing)&oldid=539260515.

[57] Z. Zhang, C. Chen, J. Sun, and K. Luk Chan. Em algorithms for gaussian mixtures with split-and-merge operation. Pattern recognition, 36(9):1973–1983, 2003.