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 2012

Resumen

Aquí va el resumen del proyecto

Descriptores

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

Abstract

Lo mismo que el resumen, pero en inglés.

Keywords

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

Índice general

Índice de figuras

Índice de tablas

Agradecimientos

Aquí irán los agradecimientos.

Esto hay que cambiarlo. Meter nuestros objetivos y tal.

El objetivo principal de este proyecto consiste en el desarrollo de una aplicación de escritorio que implemente y facilite la comprensión de los algoritmos de *Selección de Instancias clásicos*.

La selección de instancias tiene como objetivo disminuir el tamaño de las muestras para posibilitar su uso posterior. Los algoritmos de selección de instancias pretenden disminuir la complejidad mediante la reducción del número de instancias, extrayendo así las más significativas y desechando las que no aporten información al conjunto.

La necesidad de la selección de instancias se hace patente cuando se examinan los conjuntos de datos que se manejan en la vida real. Intentar entrenar un clasificador, por ejemplo, a partir de mil millones de instancias puede ser una tarea difícil e, incluso, imposible. Por ello la selección de instancias se presenta como una buena alternativa para reducir la complejidad de la muestra posibilitando su posterior tratamiento.

Actualmente los algoritmos de selección de instancias que se manejan tienen un gran inconveniente y es su complejidad. Si utilizamos la notación de Landau, el mejor de los algoritmos clásicos de selección de instancias es de complejidad $O(n^2)$ por lo que la simple tarea de seleccionar las instancias puede ser inabordable.

En este proyecto se va a implementar el algoritmo *Democratic Instance Selection* [**?**], *DIS* a partir de ahora, el cual es un nuevo algoritmo de selección de instancias basado en la técnica *divide y vencerás* y cuya complejidad es $O(n \log(n))$. Este algoritmo ha sido creado por César Ignacio García Osorio, tutor de este proyecto, y Nicolás Pedrajas García.

Con todo lo anteriormente expuesto, el desarrollo de este proyecto se presenta interesante desde dos puntos de vista: el primero es la poca orientación pedagógica de la información existente sobre los algoritmos de selección de instancias, y el segundo el desarrollo e implementación del nuevo algoritmo *DIS*. Del primer punto se extrae que el desarrollo del software debe estar diseñado para el aprendizaje de dichos algoritmos, facilitando la comprensión de los mismos y posibilitando su seguimiento. Del segundo punto se extrae que la herramienta debe tener una arquitectura pensada para ser extensible, es decir, puedan añadirse nuevos algoritmos que sigan una interfaz común.

El desarrollo se implementará sobre el lenguaje de programación *Java 6* y será capaz de ejecutar los algoritmos paso a paso, facilitando la información de las estructuras de datos que éste maneje durante su funcionamiento. Tras un análisis de alternativas, se utilizará la librería *Weka* (Waikato Environment for Knowledge Analysis) para el tratamiento y manejo de las instancias y muestras debido a su versatilidad y potencia.

Durante el desarrollo, se utilizará la metodología ágil *Scrum*. El principal motivo de escoger una metodología ágil es el desconocimiento por parte del equipo de desarrollo del entorno, haciendo imposible una planificación estructurada del mismo desde el momento del inicio del proyecto. Gracias a *Scrum* el proyecto avanzará planificándose, en cada *Sprint*, las tareas que se vayan extrayendo del *Product Backlog*.

1. OBJETIVOS DEL PROYECTO

Los objetivos principales del proyecto son principalmente cuatro:

1. Rediseñar completamente la aplicación que presentaron los alumnos del año pasado, buscando un diseño mucho más modular y reutilizable.
2. Refactorizar todo el código que reutilicemos, buscando un mejor estilo para favorecer su comprensión y reutilización.
3. Mejorar el rendimiento de la aplicación anterior, buscando que la misma se pueda aprovechar de los procesadores actuales de más de un núcleo de proceso.
4. Mejorar la precisión a la hora de localizar los defectos.
5. Ampliación de la funcionalidad de la aplicación anterior en diversos aspectos, como la implementación de algoritmos de segmentación que permitan obtener un resumen de las características geométricas de los defectos encontrados que, en un futuro, permitirán clasificar los defectos en varios y tipos y decidir si la pieza es defectuosa y debe ser retirada o si es correcta.

En el proyecto se van a implementar los algoritmos de selección 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 6*. El motivo de su elección ha sido el conocimiento del mismo, su sencillez, portabilidad, extensa documentación y la posibilidad de utilizar la librería de *Weka*. 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 *Jude*, 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 [**?**] 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 [**?**]. Las ventajas que ofrece respecto a otros sistemas de composición de textos (como los clásicos *WYSIWYG* [**?**]) 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 [**?**], manuales [**?**] y [**?**] que han facilitado el aprendizaje del mismo. Como plantilla para la memoria se utilizará la de la Universidad de Deusto [**?**] modificándola para adaptarla a la Universidad de Burgos.

Para trabajar con las radiografías se ha elegido la librería de 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 estudiarán de manera superficial los conceptos teóricos fundamentales para la correcta comprensión del proyecto. La visión por computador, también llamada visión artificial o visión técnica, es un subcampo de la inteligencia artificial. Su propósito es programar un computador para que pueda «entender» las características de una imagen. Entre sus objetivos se encuentra la detección, segmentación, localización y reconocimiento de ciertos objetos en imágenes (en nuestro caso serían defectos). Para lograrlo se utilizan diversas técnicas como el reconocimiento de patrones, aprendizaje estadístico, geometría de proyección, etc.

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.

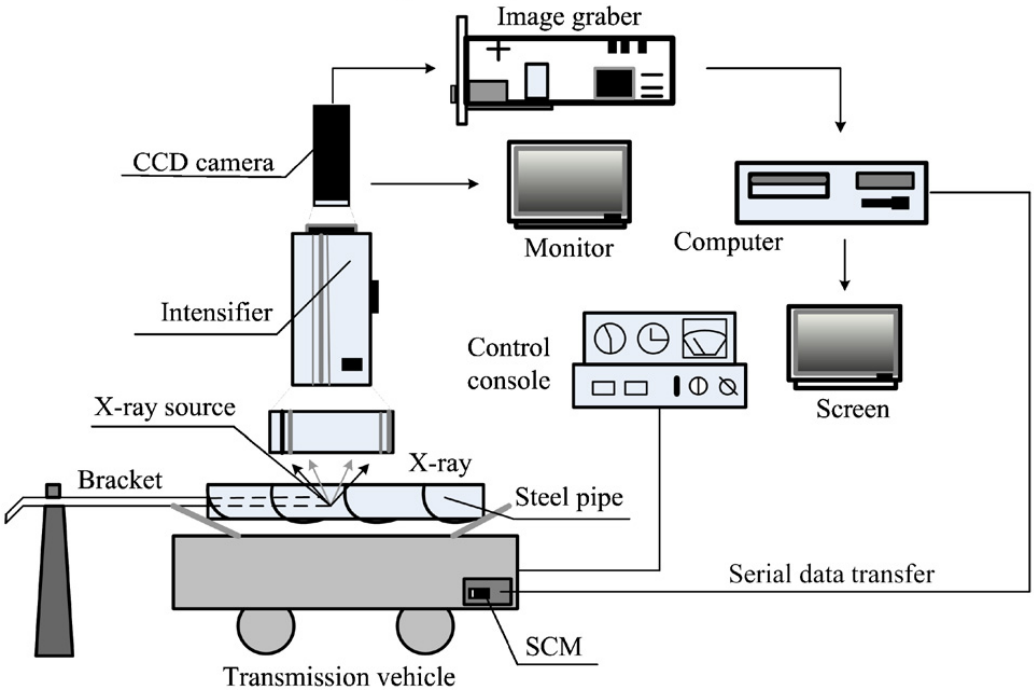

Figura 2.1: Proceso de adquisición de imágenes de radiografía

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.

2.2.1 Binarización

La binarización de una imagen consiste en un proceso mediante el cual los valores de gris de una imagen quedan reducidos a dos: verdadero y falso. 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.

2.2.2 **Filtros de detección de bordes**

HAY QUE VER CUÁLES DE ESTAS METEMOS, SI METEMOS ALGUNA

2.2.3 **Realzado de la imagen**

El realzado de imágenes es una técnica de preprocesado cuyo objetivo principal es el de destacar los detalles finos de una imagen o intensificar detalles que han sido difuminados, bien sea por error o bien por efecto natural del método de adquisición de la imagen. De esta manera, se obtiene una imagen de salida que será más fácil de interpretar, haciendo la información relevante más visible. Mediante el realzado se intenta acentuar las aristas de la imagen, obteniendo así una imagen con más contraste, es decir, con una mayor variabilidad entre los tonos de gris de sus diferentes píxeles. Con ello se consigue una mejora de la imagen, ya que los objetos aparecerán más resaltados, haciendo más fácil su diferenciación. Las utilidades del realce de las imágenes son variadas e incluyen aplicaciones que van desde la impresión electrónica y las imágenes médicas hasta las inspecciones industriales e incluso la detección autónoma de objetivos en las armas inteligentes.

2.2.4 **Saliency**

El «Saliency Map» o «Mapa de Prominencia» [43] 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. Koch y Ulman propusieron en 1985 [29] 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 [40].

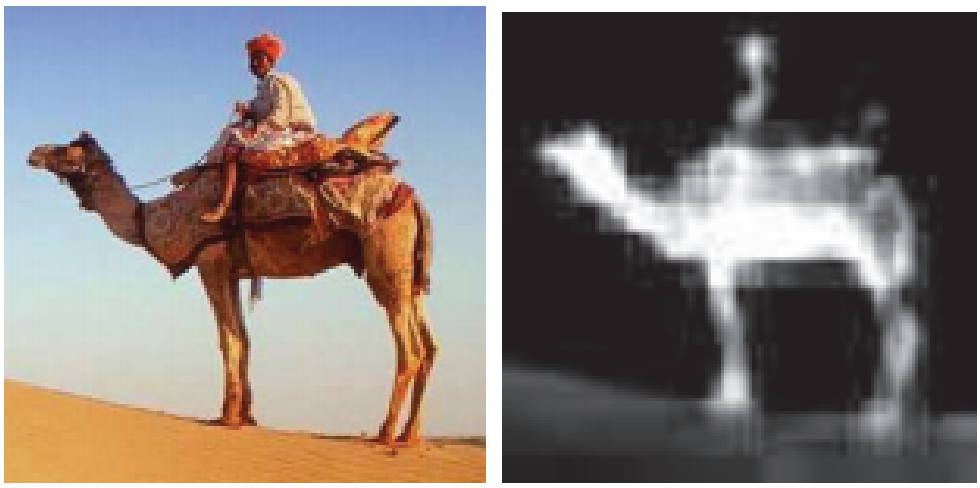

Figura 2.2: Ejemplo de Saliency Map. La imagen original a la izquierda, con el correspondiente saliency map a la derecha

2.3 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 siguientes descriptores:

2.3.1 Descriptores simples

También se les conoce como características estándar o de primer orden [46]. 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 [37] basados en aproximaciones de la primera y segunda derivada de los niveles de grises de la imagen. Ver sección 2.2.2 en la página 8. 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.

2.3.2 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 [25], 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). La forma de calcular dicha matriz está definida en el siguiente artículo [46].

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 0^o, 90^o, 180^o y 270^o 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 [2].

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

   $p(i,j)$ 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 dentro de la ventana tiene un alto contraste.

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

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:

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

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

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

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

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

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) [59] son un tipo de característica usado para la clasificación de texturas. Fueron descritos por primera vez en 1994 [52].

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 [16].

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.

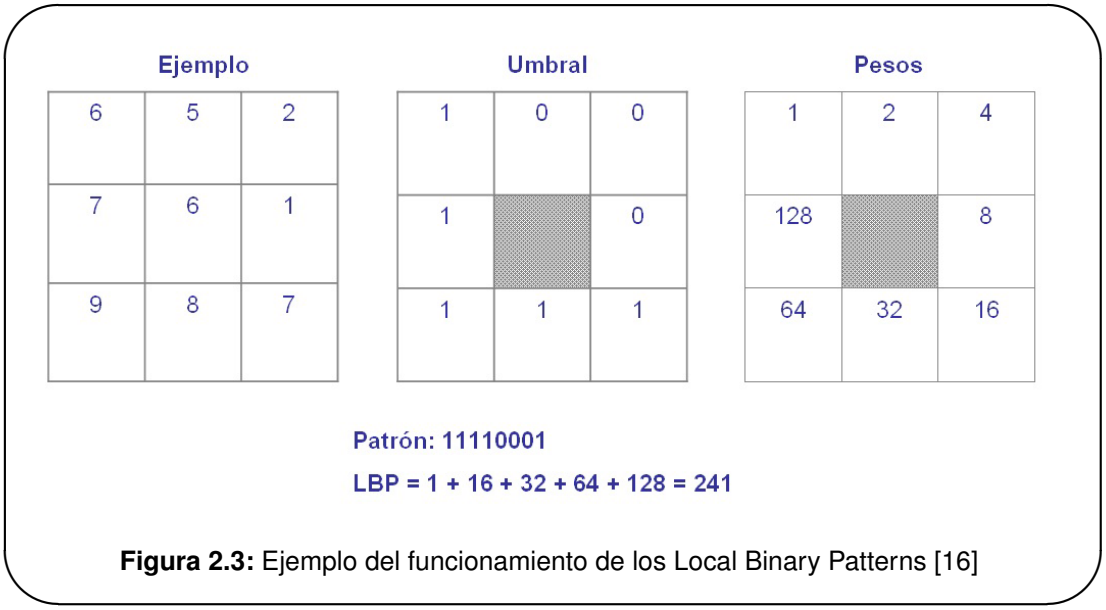

Figura 2.3: Ejemplo del funcionamiento de los Local Binary Patterns [16]

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.

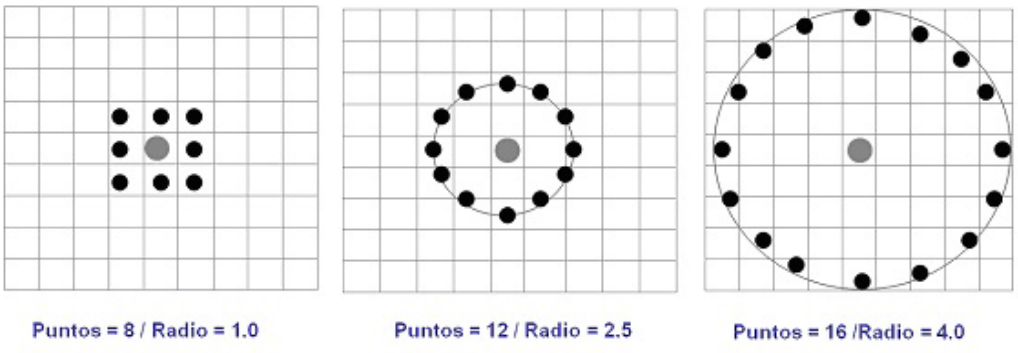

Figura 2.4: Vecindades de distinto tamaño en los LBP [16]

2.4 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 [58] es un elemento que, tomando un conjunto de características como entrada, proporciona a la salida una clase etiquetada. En nuestro caso la 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.

El clasificador no puede 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 que hicieron los alumnos del año pasado con varios tipo de clasificadores, así que hemos usado el que ellos consideraron mejor: *Bagging*. [¿INCLUIR LA INFO DE BAGGING?]

Como algoritmo base hemos usado *REPTree*, que, por sus características, también lo hemos podido usar para la regresión. [¿METER LO DE REPTREE?]

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

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

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

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

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

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