UNIVERSIDAD COMPLUTENSE DE MADRID

FACULTAD DE INFORMÁTICA
DEPARTAMENTO DE INGENIERÍA DEL
SOFTWARE E INTELIGENCIA ARTIFICIAL.

Sistemas Informáticos 2014/2015

Reconocimiento óptico de caracteres en imágenes digitales de contadores de gas

Alejandro Aparicio Martín de Loeches
Lucía Fernández Guzmán

Profesor director: Gonzalo Pajares Martinsanz

Declaración de conformidad

Los alumnos:

Alejandro Aparicio Martín de Loeches y Lucía Fernández Guzmán aquí firmantes autorizan a la Universidad Complutense de Madrid a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a sus autores, tanto la propia memoria, como el código, la documentación y/o el prototipo desarrollado.

Madrid, 1 de Junio de 2015

Alejandro Aparicio Martín de Loeches

Lucía Fernández Guzmán

RESUMEN

El reconocimiento óptico de caracteres es un proceso de digitalización de textos a partir de imágenes de símbolos o caracteres que pertenecen a un determinado alfabeto. Gracias a ello se pueden identificar y almacenar los datos reconocidos a partir de las imágenes y así poder interactuar con estos caracteres.

El presente proyecto está enfocado a este fin: obtener, a partir de las imágenes de un contador de gas, la lectura tanto del número de referencia del contador como del consumo. Se trata de una necesidad real planteada por la compañía Madrileña Red de Gas de Madrid con el fin de automatizar el proceso de lectura de contadores a la vez que se reduce el proceso de tramitación de la misma.

Se presenta una aplicación informática orientada al reconocimiento de caracteres OCR (*Optical Character Recognition)* con el fin mencionado. Lo cual constituye un avance tecnológico de interés en el ámbito empresarial.

Palabras clave: Contadores de gas, reconocimiento óptico de caracteres, OCR, imágenes digitales, procesamiento de imágenes, aplicación automática.

ABSTRACT

Optical character recognition is the conversion of characters from images.

The OCR is a process of digitization of texts from images of symbols or characters that belong to a given alphabet. Can store data such images so they can interact with these characters.

This project is focused on this goal: obtaining, from the images of a gas meter, reading both the reference number counter and consumption. Getting automate and reduce turn the meter reading process for the "Gas Company of Madrid".

We will use the abbreviation "OCR" (Optical Character Recognition) to refer to the OCR.

Keywords: Gas meters, optical carácter recognition, OCR, digital images, image processing, automatic application.

ÍNDICE

CAPÍTULO 1: Introducción

1.1 Motivación del proyecto

El reconocimiento óptico de caracteres (OCR, Optical Character Recognition) es esencial en el tratamiento de imágenes para la identificación de secuencias de caracteres con un significado concreto. Este es el caso que se plantea en el marco del presente proyecto para dar solución a un problema concreto propuesto por la compañía Madrileña Red de Gas (2015) a través de su director de operaciones de cliente el Sr. Glen Lancastle. Se pretende automatizar el proceso de lectura de contadores mediante la captura de las imágenes del contador con el fin de identificar tanto los caracteres correspondientes a la lectura que marca el consumo como el identificador de usuario.

Una imagen es capaz de transmitir de manera precisa lo ocurrido en un instante determinado. Es tan importante que se puede aplicar tanto a aspectos científicos como personales.

Desde la aparición de los algoritmos de OCR, han sido muchos los servicios que han introducido estos procesos para aumentar su rendimiento y otros que se basan completamente en estas tecnologías. Por ejemplo, el reconocimiento de texto manuscrito, reconocimiento de matrículas, indexación en bases de datos, reconocimiento de datos estructurados con OCR Zonal, son sólo algunos ejemplos dentro de otras muchas otras aplicaciones.

Además, de la motivación explícita del trabajo expresada previamente, cabe mencionar que el trabajo desarrollado ha contribuido, a través del procesamiento digital de imágenes, a conocer nuevas técnicas de tratamiento de imágenes con el

objetivo de mejorar la calidad y facilitar así la búsqueda de información en ellas, ampliando así el campo para futuras aplicaciones informáticas.

En definitiva, este proyecto está enfocado con el fin de aplicar algoritmos y diferentes técnicas de tratamiento de imágenes, para ser capaces de reconocer caracteres de interés en contadores de gas. Para ello, se ha desarrollado la aplicación que aquí se presenta bajo el lenguaje de programación JAVA.

1.2 Objetivo del proyecto

Como se ha indicado previamente, el objetivo del proyecto consiste en el desarrollo de una aplicación informática para el reconocimiento automático de secuencias de caracteres a partir de imágenes digitales en contadores de gas. Se trata de capturar una imagen e identificar los caracteres de interés en la misma, que son tanto la lectura relativa al consumo como el identificador de usuario. Esta aplicación posee especial interés para la lectura automática de contadores de gas, ampliable a otros tipos de contadores en suministros energéticos.

El presente proyecto conlleva una dificultad inherente debido a que se trata de identificar las secuencias de caracteres en imágenes reales, con la dificultad que este tipo de situaciones conlleva. En efecto, dichas imágenes proceden de contadores situados en diferentes entornos de interior y exterior, agravándose las dificultades en este último caso. Además, los dígitos a identificar proceden de una fuente poco común lo que hace todavía más difícil su interpretación. En resumen, las imágenes contienen propiedades muy difíciles de generalizar para obtener un algoritmo genérico que sirva para la lectura de todas sin fallo alguno.

1.3 Organización de la memoria

Tras el resumen previo, junto con la motivación y los objetivos del proyecto expuestos previamente, se describe a continuación brevemente la organización de la memoria.

En el segundo capítulo se analizan los conceptos teóricos en los que está basada la aplicación prototipo construida. En el tercer capítulo se detalla el diseño y la arquitectura general del prototipo. En el cuarto capítulo se explican las tecnologías utilizadas para desarrollar el proyecto, así como una explicación de por qué se han elegido. El quinto capítulo contiene un breve manual de usuario para la correcta utilización de la aplicación. El sexto capítulo explica el proceso seguido para desarrollar el proyecto. En el séptimo capítulo se detallan los resultados obtenidos para diferentes imágenes, así como las conclusiones extraídas.

CAPÍTULO 2: Conceptos teóricos

En este capítulo se introducen algunos de los conceptos teóricos y términos frecuentemente utilizados en el ámbito de la aplicación. Evidentemente, al ser un proyecto en el que se están tratando imágenes digitales, se explica dicho concepto, cómo se representan las imágenes y cuál es su contenido, así como la descripción de un histograma, que constituye un elemento esencial de los métodos de procesamiento que se incluyen en la aplicación. Además, se ilustran los distintos tipos de algoritmos utilizados para la extracción de caracteres finalizando con el concepto de binarización de imágenes.

2.1 Imágenes digitales

Las imágenes digitales son capturadas mediante los dispositivos apropiados para su almacenamiento en una computadora. La figura 1 representa un proceso de captura de una escena tridimensional. La escena 3D se proyecta, a través del sistema óptico, sobre el elemento sensor en la figura un CCD (Charge Coupled Device), que bien podría ser otro de distinta naturaleza tal como los CMOS (Complementary metal-oxide-semiconductor). En cualquier caso, de forma general, estos dispositivos poseen una distribución en forma de matriz de elementos denominados píxeles (picture elements). Cada elemento se activa con la incidencia de la radiación espectral, generando un valor de tensión eléctrica a la salida, de forma que mediante la circuitería apropiada, dicho valor se convierte a un valor numérico. Todos estos valores, organizados igualmente en forma de matriz se transfieren al computador, constituyendo lo que se conoce como imagen digital, quedando en este momento disponible para su posterior procesamiento (Pajares y Cruz, 2007).

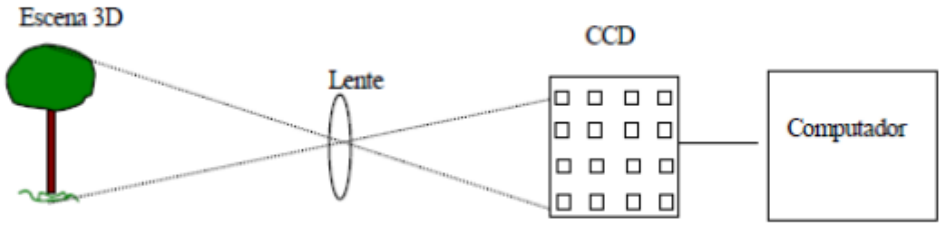


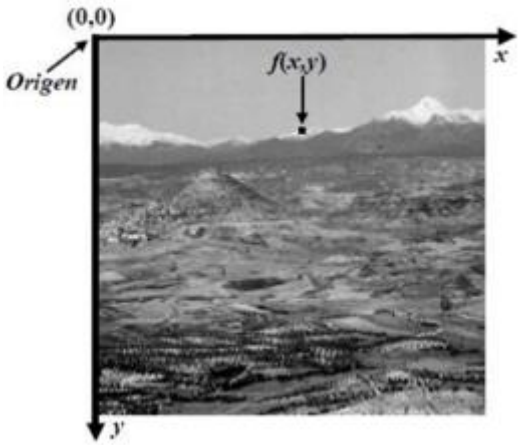
Figura 1. Captura de una imagen 3D por un dispositivo CCD.

El proceso finaliza con el almacenamiento de la imagen en el computador. Este almacenamiento se lleva a cabo a través de algún formato característico de imágenes (TIFF, BMP, JPEG).

Como se ha indicado previamente, la imagen se representa desde el punto de vista de su tratamiento computacional como una matriz numérica de dimensión MxN, es decir, con M filas y N columnas. El contenido de esa matriz se refiere a valores enteros situados en las localizaciones espaciales (*x,y*) o píxeles, dicho valor es el resultado de la cuantización de intensidad o nivel de gris.

Si la imagen es en blanco y negro, se almacena un valor por cada píxel. Este valor es el nivel de intensidad o nivel de gris comentado anteriormente. Se suele utilizar un rango de valores para su representación, que generalmente es de 0 a 2^{n-1}. Uno de los valores más utilizados es con n igual a 8; esto significa que el rango de valores para este caso varía entre 0 y 255. En este caso, el 0 representa el negro absoluto y el 255, el blanco absoluto. Esto indica que podemos tener una resolución o precisión en los grises de 256. El hecho de utilizar 256 niveles es porque con 8 bits del computador se pueden codificar 256 valores distintos desde la combinación 00000000, que representa el nivel 0, hasta la combinación 11111111, que representa el nivel 255.

Sobre cada matriz se establece un sistema de coordenadas con origen normalmente en la esquina superior izquierda y con los ejes x e y tales que su orientación es positivo hacia la derecha en el caso del eje X y hacia abajo en el caso del eje Y, figura 2.



$$I(x,y) = \begin{bmatrix} 2 & 5 & 8 & 0 \\ 2 & 2 & 9 & 1 \\ 2 & 4 & 6 & 3 \\ 2 & 4 & 4 & 4 \end{bmatrix}$$

Figura 2. Representación de una imagen digital en grises.

Con esta disposición de imágenes se tiene acceso a los valores de los píxeles en sus diferentes localizaciones, siendo el convenio de acceso el que se expresa seguidamente con relación a la figura 2.2:

$$I(x,y) = I(2,1) = 5$$

Cuando la imagen es en color, para cada localización espacial existen tres valores de intensidad asociados, es decir, los elementos de la matriz vienen dados por tres valores que representan cada uno de los componentes básicos del color en cuestión. Estos componentes son el rojo (R), verde (G), y azul (B) y es lo que

conocemos como código RGB. En este caso el conjunto de valores (0, 0, 0) representa al negro, mientras que los valores (255, 255, 255) es el blanco absoluto. La combinación de distintos valores representa otros colores. Debido a lo anterior, una imagen en color posee tres bandas espectrales: rojo, verde, azul; cada una de ellas viene representada por una matriz de números con valores en el rango 0 a 255 para imágenes de 8 bits. Una imagen de color vendría dada por las tres subimágenes de la figura 3, Martínez, L. (2015).

$$R(x,y) = \begin{bmatrix} 2 & 5 & 8 & 0 \\ 2 & 2 & 9 & 1 \\ 2 & 4 & 6 & 3 \\ 2 & 4 & 4 & 4 \end{bmatrix}; \quad G(x,y) = \begin{bmatrix} 8 & 4 & 4 & 3 \\ 8 & 3 & 4 & 0 \\ 6 & 9 & 1 & 6 \\ 6 & 9 & 3 & 5 \end{bmatrix}; \quad B(x,y) = \begin{bmatrix} 7 & 6 & 7 & 3 \\ 7 & 6 & 8 & 4 \\ 1 & 9 & 7 & 2 \\ 3 & 9 & 9 & 6 \end{bmatrix}$$

Figura 3. Representación de una imagen digital en color:
Rojo(R), Verde (G) y Azul (B).

El píxel que se encuentra en la localización espacial (x,y) posee componentes (R, G, B). Por ejemplo, dada la imagen de la figura 2.3 los valores del píxel en $(x,y) = (2,3)$ serían los siguientes: (4, 9, 9). Los tres valores de cada píxel para las imágenes en color corresponden al modelo de color RGB.

2.2 El histograma

El histograma de una imagen es una función discreta que representa el número de píxeles en la imagen en función de los niveles de intensidad, g. La probabilidad de ocurrencia de un determinado nivel se define como:

$$P(g) = \frac{N(g)}{M}$$

Donde M es el número de píxeles en la imagen y N(g) es el número de píxeles en el nivel de intensidad g. Como con cualquier distribución de probabilidad todos los

valores de P(g) son menores o iguales que 1 y la suma de todos los valores de P(g) es 1.

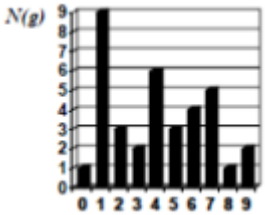
El histograma de una imagen y su representación constituye una herramienta visual de gran utilidad para el estudio de imágenes digitales, que puede proporcionar una idea muy aproximada de la distribución de niveles de gris.

Las intensidades o niveles de gris están representadas a lo largo del eje X y el número de ocurrencias para cada intensidad se representa en el eje Y.

Por ejemplo, dada la imagen de la figura 4 (a) con 10 niveles de gris, del 0 al 9; su histograma se muestra en (b).

$$\begin{bmatrix} 0 & 3 & 3 & 4 & 4 & 4 \\ 1 & 1 & 1 & 4 & 4 & 5 \\ 1 & 1 & 1 & 4 & 5 & 5 \\ 1 & 1 & 1 & 7 & 6 & 6 \\ 2 & 2 & 7 & 7 & 6 & 6 \\ 8 & 2 & 7 & 7 & 9 & 9 \end{bmatrix}$$

(a)



(b) Niveles de gris

Figura 4. (a) Imagen con 10 niveles de gris del 0 al 9; (b) Su histograma.

En la figura 5 (a) se muestra una imagen de grises relativa a un contador de los analizados en el proyecto y en (b) su histograma de los niveles de gris en cada uno de los tres canales con valores de intensidad variando en el rango del 0 al 255. La imagen mostrada es de grises, de forma que cuando se almacena en cualquiera de los tres formatos indicados previamente, la misma imagen se triplica de forma que cada canal R, G y B representa exactamente la misma imagen, de ahí que los tres histogramas mostrados en la figura 6 sean exactamente los mismos.

Figura 5. Imagen original de grises.

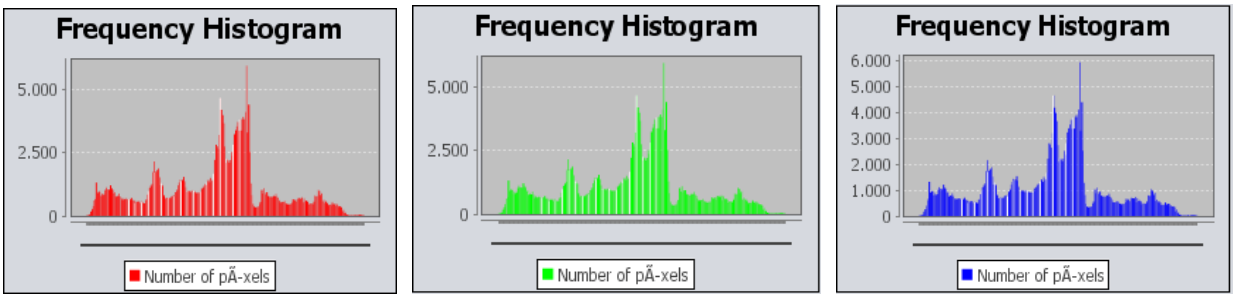


Figura 6. Histogramas de los tres canales espectrales en el modelo de color RGB.

2.3 Algoritmo de detección de rojos

La parte numérica decimal de los contadores que han servido de base para el desarrollo del presente proyecto aparece en rojo. Esta circunstancia permitirá la localización de una de las zonas numéricas de interés para su posterior análisis.

Con tal propósito, se ha desarrollado un algoritmo que nos servirá como estrategia para la localización de la parte roja de una imagen original. Nos centraremos en localizar esta parte del contador para tener una orientación medianamente precisa a la hora de localizar el número de referencia del contador y su consumo ya que ambos se encuentran en la parte superior e izquierda respectivamente de la mencionada zona roja.

2.3.1 Localización de rojos

Se realiza un recorrido sobre los píxeles de la imagen obteniendo los valores RGB de cada uno de ellos calculando el porcentaje de rojo sobre verde (PRG) y rojo sobre azul (PRB) como se expresa a continuación:

$$PRG = \frac{R}{G+\varepsilon}; \qquad PRB = \frac{R}{B+\varepsilon} \tag{1}$$

Donde $\varepsilon = 10^{-16}$ para evitar divisiones por cero.

Finalmente, para localizar con exactitud la zona requerida se utiliza la siguiente regla de decisión: "un píxel se marca como rojo si PRG y PRB son ambos simultáneamente mayores que la unidad".

En la figura 7 se muestra una imagen con el resultado obtenido tras la aplicación de la regla de decisión anterior a la imagen original El color rojo localizado se ha etiquetado como verde para poder ver así los píxeles que se reconocen como rojo en su mayoría de componentes.

Figura 7. Localización de parte roja del contador.

2.3.2 Normalización del espacio de color

Para llevar a cabo el estudio de la zona roja del contador es necesario realizar una normalización del espacio del color. Partimos inicialmente de una imagen que se encuentra en el espacio de color RGB, con las tres consabidas componentes espectrales R, G, B variando en el rango 0 a 255. A cada uno de los píxeles se les aplica el siguiente proceso de transformación en los tres canales:

$$\text{Rojo: } r = \frac{R}{R+G+B}; \quad \text{Verde: } g = \frac{G}{R+G+B}; \quad \text{Azul: } a = \frac{B}{R+G+B} \tag{2}$$

Donde R, G y B se obtienen de la siguiente forma:

$$R = \frac{R}{Rmax} \qquad G = \frac{G}{Gmax} \quad B = \frac{B}{Bmax} \tag{3}$$

Donde Rmax, Gmax, y Bmax representan los valores máximos en cada uno de los canales respectivos, normalmente vienen a ser 255, aunque no necesariamente.

2.4 Binarización de imágenes

La binarización de una imagen digital consiste en transformar la imagen en escala de grises en una imagen en blanco y negro. Para realizar la operación de binarización, se deberá elegir un valor adecuado de umbral dentro de los niveles de grises. Una vez elegido el umbral, todos los niveles de grises menores que el valor de umbral fijado se convertirán en negro y todos los que resulten ser mayores en blanco.

$$g(x,y) = \begin{cases} 0 \ \ si\ f(x,y) < T \\ 1\ si\ f(x,y) \geq T \end{cases} \tag{4}$$

En nuestro caso, el proceso de binarización transformará las partes verdes de la imagen a blanco y el resto a negro. El punto clave es seleccionar un umbral óptimo, es decir, un valor de gris a partir del cual se decide cuáles de dichas tonalidades son o no verdes.

Para realizar la binarización nos basaremos en el histograma de la imagen, que constituye el punto de partida del proceso.

Existen diferentes métodos de binarización de imágenes. De entre ellos, el método de Otsu (1979) es uno de los más clásicos y populares. Otro método posible es la binarización mediante la media estadística del histograma como valor de umbral.

2.4.1 Binarización mediante el método de Otsu

El método de Otsu resulta ser un método de binarización de imágenes digitales ampliamente utilizado en la literatura. Se basa en conceptos estadísticos, en concreto la dispersión de valores (en este caso, niveles de gris) a través de la varianza, Bostjan (2015). Si partimos de una imagen con L niveles de intensidad, en nuestro caso 256, y asumiendo que el umbral buscado es T, las probabilidades hasta T y desde T hasta L resultan ser:

$$w1(t) = \sum_{z=1}^{T} P(z), \qquad w2(t) = \sum_{z=T+1}^{L} P(z) \tag{5}$$

Donde $P(z)$ es el total de apariciones del nivel de intensidad z. A continuación, se obtienen las medias y las varianzas asociadas:

$$\sigma 1(t)^2 = \sum_{z=1}^{T} (z - \mu 1(t))^2 \frac{P(z)}{w1(t)}; \;\; \sigma 2(t)^2 = \sum_{z=T+1}^{L} (z - \mu 2(t))^2 \frac{P(z)}{w2(t)} \tag{6}$$

Finalmente, se obtiene la varianza ponderada:

$$\sigma_w^2(t) = w1(t)\sigma1(t)^2 + w2(t)\sigma2(t)^2 \quad (7)$$

Se elige el umbral T correspondiente al nivel de intensidad que proporcione la mínima varianza ponderada.

2.4.2 Binarización mediante la media

Por media se entiende el valor medio de los niveles de intensidad en la imagen. Se obtiene directamente a partir del histograma de la imagen, mediante la siguiente ecuación:

(8)

$$m = \sum_{i=0}^{L-1} z_i * p(z_i)$$

Donde L es el número total de niveles de intensidad, en nuestro caso 256. Las imágenes mostradas en la figura 8 son el resultado de aplicar los mencionados métodos de binarización.

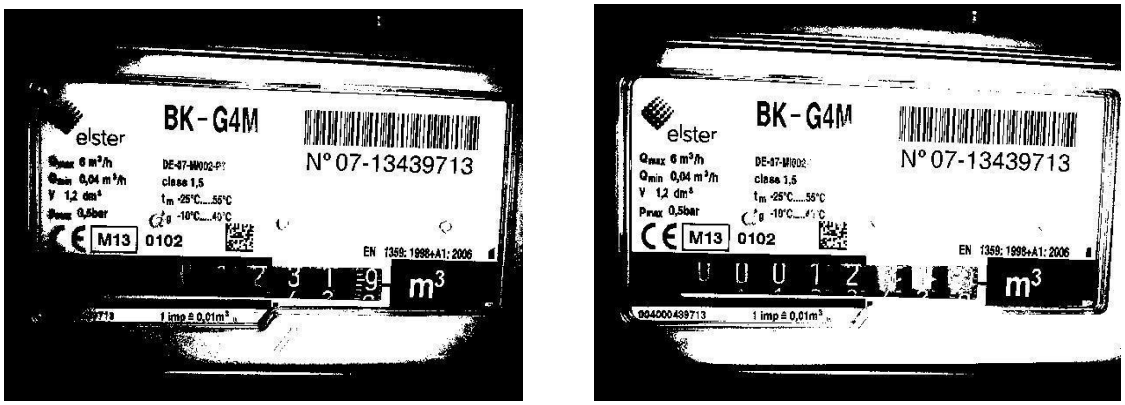


Figura 8. Imágenes binarizadas con la media (izquierda) y Otsu (derecha).

CAPÍTULO 3: Fase de análisis

Este capítulo tiene por objetivo la explicación y documentación de las fases típicas de todo proyecto de desarrollo de software, que son las aplicadas en nuestro caso:

- **Fase de requisitos**, en la cual se identificarán los requisitos del sistema
- **Fase de diseño,** es esta fase se detallarán los aspectos más importantes de la arquitectura del sistema
- **Fase de implementación,** en la que se traduce el diseño a un lenguaje de programación (en nuestro caso JAVA), generación de código e implementación de funciones como tareas principales.
- **Fase de pruebas,** como en todo proyecto, se realizan diversas pruebas software para comprobar si se obtiene el resultado esperado.
- **Fase de mantenimiento,** orientado a mantener la operatividad del software, ya que al utilizarlo como usuario final puede ser que no cumpla con todas las expectativas previstas.

Todas las acciones y tareas que se incluyen y se realizan sobre cada una de estas fases, es lo que se denomina como el ciclo de vida de un proyecto software.

Existen varios modelos para explicar el ciclo de vida de un proyecto de estas características. Uno de los más comunes es sin duda el relativo al desarrollo en cascada, figura 9. Es una visión del proceso de desarrollo software como una sucesión de fases que producen productos intermedios y al final de cada una de estas fases se deben revisar las tareas y productos que se han generado en la fase correspondiente. Es decir, cada etapa deja el camino preparado para la siguiente,

de forma que esta última no puede comenzar hasta que no haya terminado la anterior.

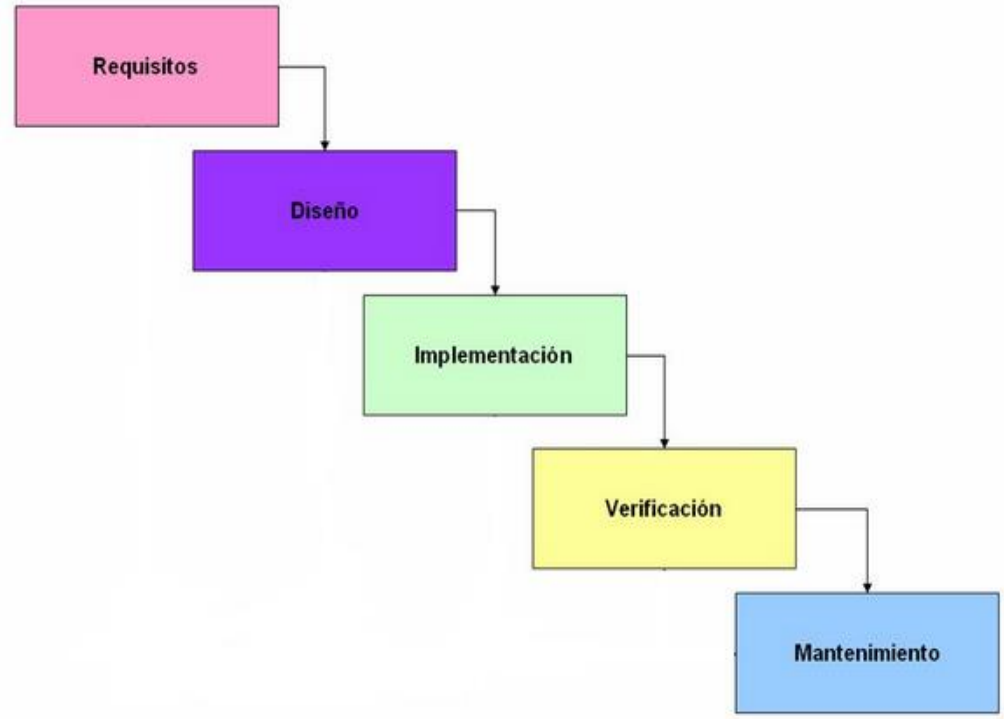


Figura 9. Modelo en cascada del desarrollo software.

La principal limitación es que no se permiten hacer iteraciones entre las fases y por tanto resulta poco realista desde un punto de vista práctico. De estos problemas surgen las principales variantes a este modelo, como por ejemplo el modelo en cascada con retroalimentación entre fases, figura 10. Si al terminar una etapa los resultados no son los esperados, podemos volver a la fase anterior. Este modelo es mucho más flexible y realista en la práctica que el anterior. Además, promueve una metodología de trabajo efectiva basada en los siguientes principios: definir antes de diseñar, diseñar antes de implementar.

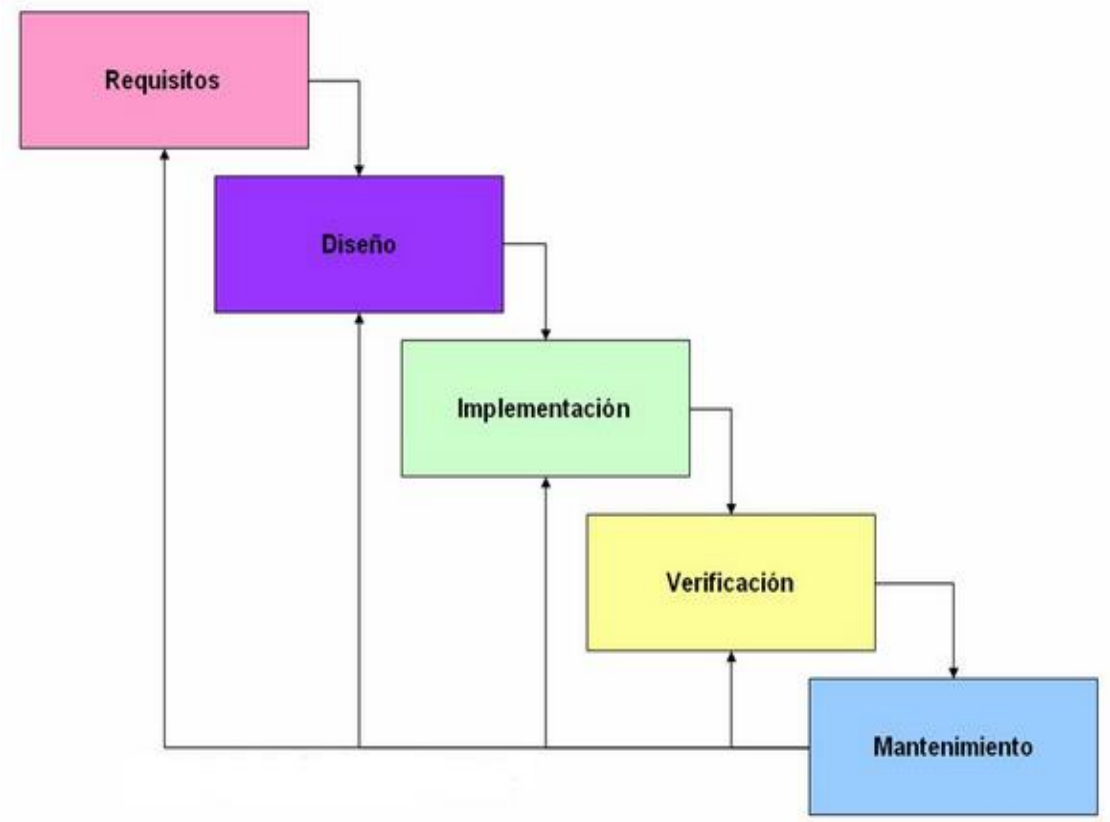


Figura 10. Modelo en cascada del desarrollo software con retroalimentación.

Por tanto, en vista de las ventajas y desventajas de ambos modelos, hemos elegido el modelo en cascada con retroalimentación a la hora de desarrollar nuestro proyecto software. En las etapas de análisis y diseño se emplean diagramas UML, Arlow y Neustadt (2006), utilizados para definir, documentar y construir un sistema.

Los requisitos que se explican a continuación están relacionados directamente con las propiedades que debe tener el funcionamiento de la aplicación software.

Facilidad para la ejecución paso a paso.

Para lograr que la aplicación sea lo más sencilla y sobre todo didáctica posible trataremos de que el algoritmo del programa se pueda ir ejecutando paso por paso, identificándose claramente en cada uno de estos pasos que se está haciendo y por qué.

Facilidad de uso de la aplicación

Pretendemos que nuestra aplicación sea lo más fácil posible de utilizar prescindiendo de interfaces complicadas y recargadas de botones. Mientras menos botones más facilidad de uso. Además, éstos serán muy intuitivos desde un punto de vista práctico.

Cálculos complejos transparentes al usuario

Tratamos que la aplicación sea lo más visual posible, por lo que los cálculos o tratamientos de la imagen junto con sus algoritmos deben quedar lo más posible ocultos al usuario. Al usuario final del sistema sólo le interesa que el sistema reconozca el consumo del contador de gas y su número de referencia. El cómo se llega al reconocimiento de los caracteres debe quedar oculto porque no le interesará, además de que complicaría la interfaz y el uso del programa.

Rapidez de reconocimiento

Los algoritmos a usar deben ser lo menos complejos temporalmente posible, ese es un objetivo importante, lograr una aplicación lo más rápida ágil en el reconocimiento mejor, para que pueda usarse en la medida de lo posible en sistemas en tiempo real con una alta carga de trabajo.

Robustez del algoritmo

Este es el gran caballo de batalla de todos los reconocedores de imágenes, en la medida de lo posible el algoritmo debe ser robusto ante posibles deficiencias en la fotografía original, debidas en gran parte a los problemas derivados de la captura de las imágenes en entornos de exterior. Sin embargo hay que decir que no todas las posibles dificultades son salvables, una muy baja luminosidad o un alto ruido, entre otras cosas, pueden hacer que hasta el mejor reconocedor falle o incluso no funcione.

El resto de requisitos que se explican a continuación se refieren a propiedades sobre la calidad que debe tener el producto software final.

Escalabilidad

El sistema podría ser ampliado con facilidad ya que el tratamiento de estas imágenes no es completamente efectivo. Por ejemplo la funcionalidad de dicho sistema a partir de diferentes perspectivas angulares.

Modularidad

Es recomendable que el programa esté dividido en módulos más o menos independientes de manera que cada uno de ellos realice tareas específicas y concretas. De esta forma si se necesita mejorar o corregir un aspecto del programa sólo tendríamos que modificar aquellos módulos implicados sin necesidad de intervenir en todo el programa. Por lo tanto se logra también un mejor mantenimiento del sistema.

Usabilidad

Se trata de que la aplicación pueda ser utilizada por todo tipo de usuarios, desde el usuario experto en la tecnología hasta el usuario más novel.

Fiabilidad

El sistema debe dar los resultados lo más correctos posibles. En nuestro caso, no podemos asegurar el correcto funcionamiento del sistema ya que una detección del consumo o número defectuosa daría lugar a problemas para la empresa y sus clientes.

Bajo coste computacional y de almacenamiento

Está claro que nos interesa unos algoritmos de coste computacional bajo, tanto espacial como temporalmente, ya que al proporcionarle una determinada funcionalidad, en la práctica interesa que los algoritmos se computen lo más rápido posible ocupando el mínimo especio en memoria para poder ejecutar la aplicación de forma efectiva.

CAPÍTULO 4: Fase de diseño

Después de la fase de análisis y determinar las distintas propiedades o requisitos que debería tener el sistema a desarrollar, pasamos a la fase de diseño. En esta fase se realiza una propuesta de cómo debe ser la lectura del contador de gas a nivel de diseño, que lógicamente vendrá guiado por la fase de análisis anteriormente expuesta. La estrategia de la solución de diseño se centra en la creación de un programa capaz de reconocer los caracteres del contador a partir de una imagen fotográfica del contador en su entorno real. Se pretende que la solución sea lo más fácil posible de usar, por lo que se intenta automatizar y hacer lo más simple posible las operaciones a realizar por la aplicación, para alcanzar la funcionalidad y solución deseadas.

4.1 Vista Conceptual

La arquitectura para el diseño de la aplicación debe estar cimentada en los requisitos funcionales y no funcionales descritos en la fase de análisis, es decir la arquitectura debe diseñarse para lograr el objetivo final, esto es, reconocer los números de interés del contador de gas. En la figura 11 podemos distinguir la siguiente vista conceptual teniendo en cuenta la visión que el usuario tendría de la aplicación.

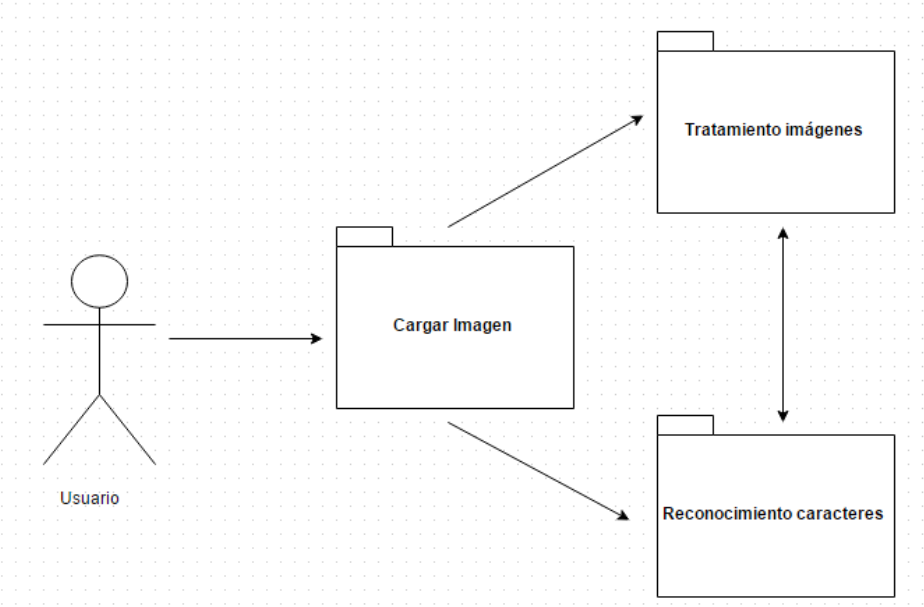


Figura 11. Vista conceptual.

Las funciones básicas y los detalles de estos subsistemas se describirán más adelante en el apartado 4.4, diagrama de Clases.

4.2 Vista Lógica

En lo que se refiere a la vista lógica, la aplicación, como puede verse en la vista conceptual, presenta dos subsistemas recogidos en un único sistema que proporciona la funcionalidad de los algoritmos para el tratamiento de imágenes digitales.

Se tiene por un lado la ***carga de imagen***, que será una fotografía que cargaremos directamente al sistema, ya que como se ha informado anteriormente, la aplicación software no se encarga de la extracción de la foto como tal. Por otro lado, aparece el ***tratamiento de imágenes***, el cual se encargará de realizar los algoritmos para poder localizar las zonas de interés del contador. Por último,

aparece el ***Reconocimiento de caracteres*** del contador; una vez localizadas las zonas requeridas, se procederá a la lectura de dichos caracteres. En la figura 12 se muestra el esquema general de la vista lógica.

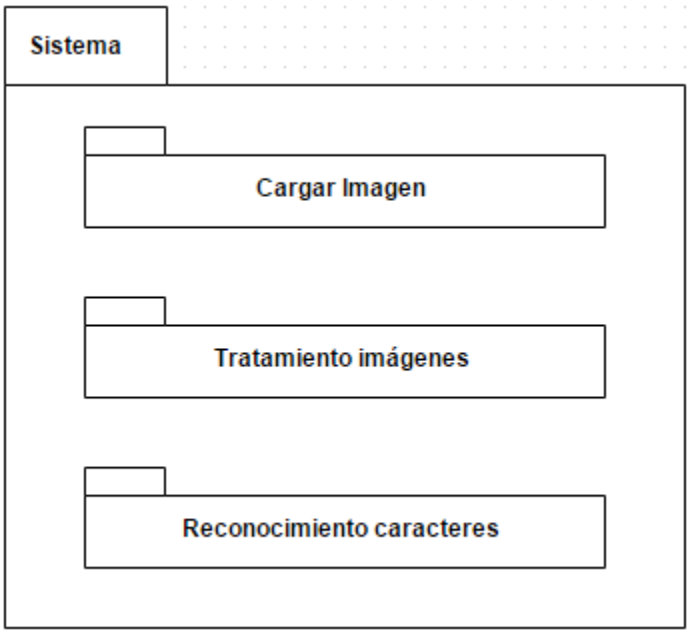


Figura 12. Vista lógica.

4.3 Arquitectura Modelo-Vista-Controlador

La arquitectura Modelo-Vista-Controlador (MVC) es un patrón de diseño que separa la lógica de negocio, la interfaz de usuario y los datos con los que se trabaja. Divide la aplicación en tres partes:

- Un modelo que contiene la información con la que se trabaja, es decir, los datos.
- Vistas que gestionan cómo se muestra la información al usuario de una manera accesible.
- Uno o varios controladores que realizan las funciones correspondientes en el modelo como respuesta a eventos procedentes generalmente de la vista.

Una ventaja de usar el MVC es que se incrementa el nivel de reutilización de los diferentes módulos software y de la aplicación en general. Para ello el modelo no puede tener acceso a ninguna clase del Controlador ni de la Vista. Además, el cambio de modelo no debe afectar a las vistas, y el controlador debe ver las clases del modelo, pero no de la vista. En la figura 13 pueden observarse las principales características de este patrón, así como la manera de interactuar entre los distintos componentes que forman parte de la arquitectura MVC.

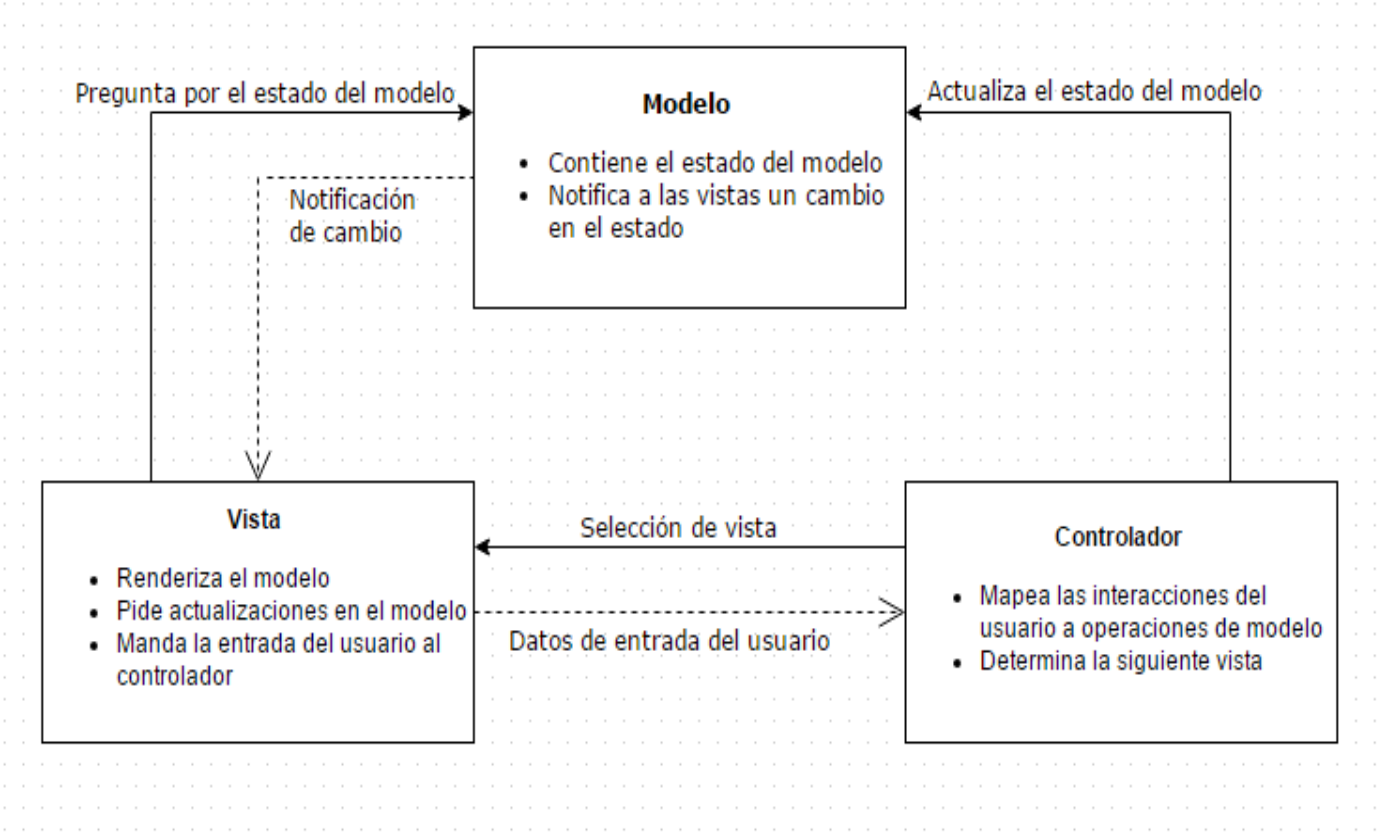


Figura 13. Arquitectura modelo-vista-controlador.

El uso del patrón MVC permite la evolución de forma separada de cada una de las partes, lo que proporciona una mayor flexibilidad a la hora de desarrollar los componentes del sistema.

Aunque existen diferentes implementaciones de MVC, normalmente el flujo que sigue es el siguiente:

- Usuario interactúa con la interfaz de usuario.
- El controlador gestiona el evento de entrada.
- El controlador informa al modelo sobre la acción elegida por el usuario, pudiendo acceder al modelo y modificarlo si fuera necesario de acuerdo a dicha acción.
- Se modifica la vista, tomando los datos del modelo.
- La interfaz de usuario espera nuevas interacciones del usuario.

Las ventajas sobre el uso de MVC son las que expresan a continuación:

- Teniendo un mismo modelo, es posible conservar diferentes vistas.
- Es posible añadir nuevas vistas sin la necesidad de modificar el modelo
- Menor acoplamiento entre vistas y modelo.
- Diseño más claro.

Evidentemente, este modelo no es único, existen muchas variaciones e interpretaciones, si bien, lo importante es separar los tres elementos de que consta, algo que cumple el utilizado en este trabajo.

4.3.1 Diseño del módulo

A continuación, se muestra en la figura 14, el módulo que se encarga de extraer el área de la fotografía que delimita las zonas de interés. Su entrada es la imagen digital del contador y su salida son los números que hemos sido capaces de reconocer. En los pasos intermedios se indican las operaciones a realizar, exactamente con el nombre con el que aparecen en la figura,

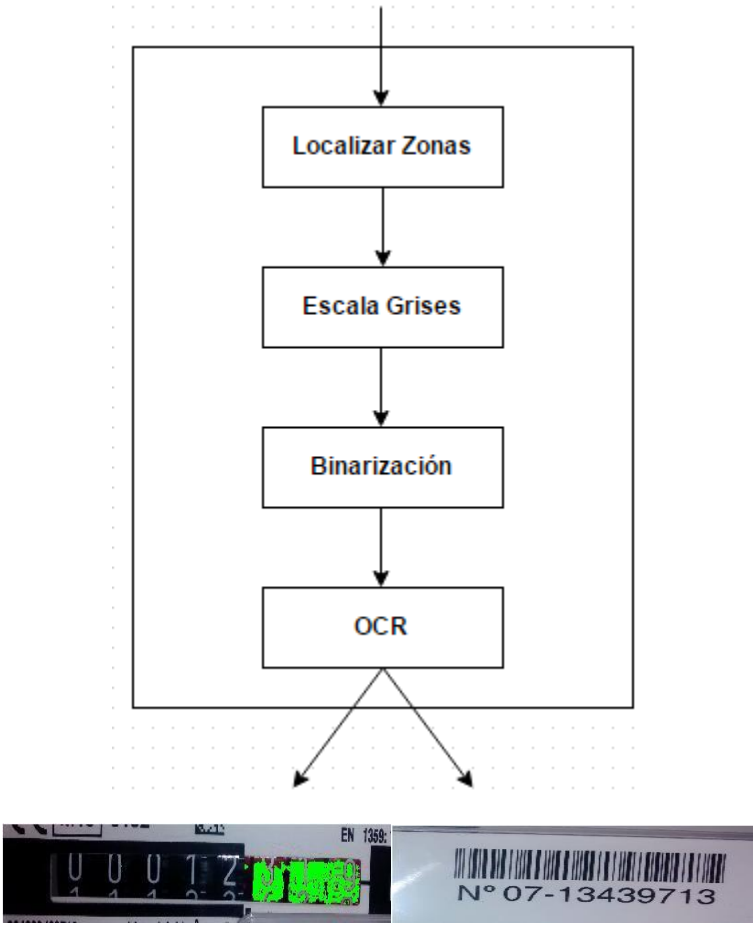


Figura 14. Diseño del módulo.

4.4 Diagrama de clases

El diagrama de clases es el diagrama principal para el diseño estático, presentando y describiendo las clases y objetos del sistema con sus relaciones estructurales y de herencia. En la definición de clases se muestran los atributos y métodos incluidos en la clase correspondiente.

En el esquema de la figura 15 aparece representada la organización y las clases del paquete *contador*, usando para ello el diagrama de clases del mencionado paquete. En dicha figura se presenta el modelado de las clases que nos permitirá

generar los algoritmos para la localización de las zonas requeridas. Contiene la clase abstracta *ProcesamientoImagen*, siendo la clase que abarca todas las funcionalidades necesarias para implementar los métodos para el tratamiento de las imágenes. También contiene la clase *Interfaz*, la cual, se encargará exclusivamente de la presentación visual de la aplicación.

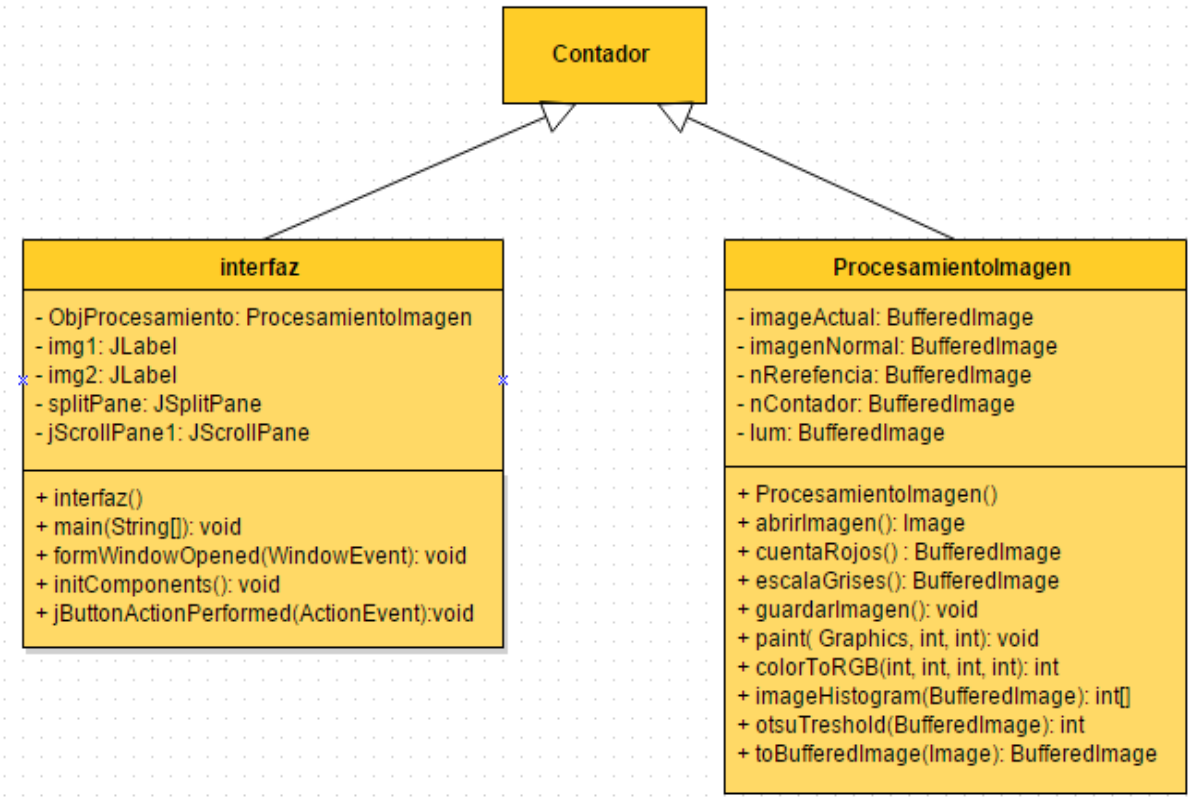


Figura 15. Diagrama de paquetes.

4.5 Casos de uso

Como parte final a la fase de diseño se encuentra el diagrama de casos de uso, el cual describe detalladamente el funcionamiento de la aplicación visto tanto desde la perspectiva del usuario como del sistema, que se identifican como dos actores presentes en el sistema, figura 16.

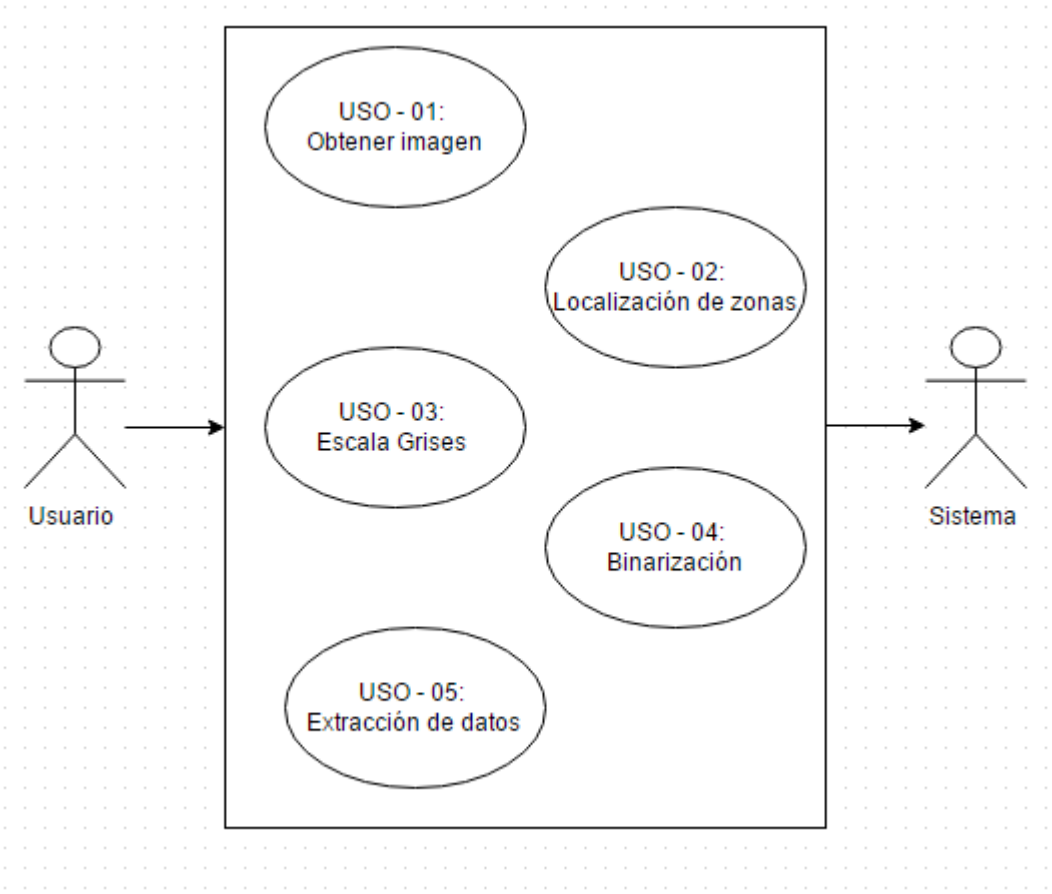


Figura 16. Diagrama casos de uso.

CASOS DE USO:

USO - 01: Obtener imagen

Descripción: paso previo al reconocimiento, consiste en cargar la imagen del contador.

1. Usuario abre Interfaz y pulsa sobre el botón "Cargar Imagen".
2. El sistema muestra a la izquierda del interfaz la imagen y se prepara para el tratamiento.
3. Paso al caso de uso siguiente: "Localización de zonas (USO-02)".
4. El caso de uso termina.

USO - 02: Localización de zonas

Descripción: primera fase del algoritmo de tratamiento de imágenes, buscamos el área donde se encuentra la zona de consumo así como la zona del número de referencia del contador.

1. Usuario solicita al Sistema la localización de las zonas pulsando en el interfaz sobre el botón "Localiza zonas".
2. Sistema realiza paso 1 del algoritmo: localiza las zonas deseadas y las muestra a la derecha del interfaz.
3. Paso al caso de uso siguiente: Escala Grises (USO - 03).
4. El caso de uso termina.

USO - 03: Escala Grises

Descripción: es la segunda fase de la aplicación, ya que la primera fase es la localización de las zonas, ahora procede realizar el tratamiento de la imagen.

1. Usuario solicita al Sistema hacer el escalado a grises pulsando en el interfaz sobre el botón "Escala grises".
2. Sistema realiza el paso 2 del algoritmo: convierte el valor de los píxeles en una graduación equivalente a gris. Muestra el resultado a la derecha de la interfaz.
3. Paso al caso de uso siguiente: Binarización (USO - 04).
4. El caso de uso termina.

USO - 04: Binarización

Descripción: es la tercera fase de la aplicación, en este caso realizaremos una segmentación de la imagen en dos partes:

1. Usuario solicita al Sistema realizar la binarización de la imagen pulsando en el interfaz sobre el botón "Binarizar".

2. Sistema realiza el siguiente paso del algoritmo, se encarga de mostrar la imagen binarizada a la derecha de la interfaz.
3. Paso al caso de uso siguiente: Extracción de datos(USO - 05)
4. El caso de uso termina.

USO - 05 Extracción de datos

Descripción: en esta última fase, aplica el algoritmo OCR sobre las zonas deseadas y se muestran los resultados obtenidos.

1. Usuario solicita al Sistema hacer el OCR así como obtener los resultados de la imagen pulsando en el interfaz sobre el botón "OCR".
2. Sistema realiza el último paso: realiza el algoritmo OCR sobre el número de referencia y su consumo y muestra por pantalla los datos resultantes.
3. El caso de uso termina.

CAPÍTULO 5: Fase de Implementación

5.1 Descripción

En esta fase se trata todo lo relacionado con la implementación de la aplicación, se verán las técnicas que hemos usado para la extracción de los caracteres y las técnicas utilizadas para el reconocimiento de los mismos. También veremos globalmente cuál es la estructura de la aplicación y las distintas fases que la componen. Todo este proceso de implementación se adecua a todo lo desarrollado en la fase de análisis y en la fase de diseño.

5.2 Lenguaje de Programación: Java

El hecho de que sea un lenguaje multiplataforma y el haber tenido suficiente contacto con este lenguaje a lo largo de estos últimos años fue decisivo a la hora de elegirlo como lenguaje de desarrollo. Aunque presenta algunas desventajas para esta aplicación en concreto, posee otras muchas ventajas que superan a las negativas. Son la que se enumeran a continuación:

- **Orientado a objetos**

Se facilita mucho el uso de técnicas como la modularidad, herencia, abstracción, polimorfismo y encapsulamiento, las cuales se resultan muy útiles para los objetivos, sobre todo de cara a una futura evolución y ampliación de la aplicación.

- **Open Source**

Gran flexibilidad y que sea capaz de evolucionar según las necesidades de mercado y usuarios constituyen una gran ventaja.

- **Independiente de la plataforma**

El hecho de que la aplicación sea ejecutable independientemente del sistema operativo o la arquitectura de la máquina.

- **Recolector de basura**

Esta característica está muy ligada a la orientación a objetos. En java, es el propio lenguaje el que se encarga de destruir automáticamente los objetos que se han quedado sin referencia, liberando así la memoria. Además, en nuestro caso particular, tratamos con un gran número de datos así que esta característica se vuelve esencial.

- **Facilidad de aprendizaje**

Java es un lenguaje muy sencillo de aprender y muy manejable para enfrentarse a nuevos retos. Su gran comunidad de programadores a nivel mundial permite ampliar rápidamente conocimientos en el caso de ser necesario ya que hacen que resulte fácil encontrar información y ayuda.

- **Entornos de desarrollo**

Al ser un lenguaje tan utilizado posee muchas alternativas a la hora de elegir con qué herramienta queremos trabajar. Además, casi todas ellas son accesibles y sencillas de utilizar. En nuestro caso hemos elegido Eclipse por su facilidad a la hora de diseñar interfaces a la vez que resulta ser de amplia difusión.

- **Librerías**

Existe una gran variedad de librerías que facilitan el tratamiento de operaciones y por tanto el trabajo de desarrollo. El poder trabajar con estas librerías facilita enormemente el trabajo de implementación.

5.3 Librerías

Como se ha mencionado previamente, Java ofrece gran variedad de librerías internas de gran utilidad, pero desgraciadamente no ofrecían determinadas funcionalidades que para nuestro desarrollo eran imprescindibles. A continuación, se explican en detalle las librerías externas añadidas a proyecto, Laganière, R. (2011).

5.3.1 JMathPlot

JMathPlot es una biblioteca de gráficos 2D y 3D de libre distribución que facilita enormemente las representaciones gráficas. Aunque esta librería no tiene un uso de cara al usuario sí que nos ha sido muy útil a la hora de representar los histogramas de una manera clara, pudiendo así observar con facilidad el comportamiento de las imágenes.

5.3.2 Imshow

Como Java no es un lenguaje específico para el tratamiento de imágenes hemos tenido que adquirir esta librería para una mayor facilidad a la hora de movernos con ellas. Sin Imshow nos encontramos con muchos problemas a la hora de mostrar la imagen por pantalla y ver si estábamos realizando bien el tratamiento. Además, también nos sirvió de gran ayuda para la manipulación de las propias imágenes.

5.3.3 OCR

La biblioteca escogida para la realización de OCR se llama ASPRISE Java OCR (2015) El proceso que lleva a cabo esta librería consiste en extraer unos caracteres, en este caso, las secuencias numéricas de caracteres del contador, que aparecen en una imagen en un archivo de texto que podrá ser editado y utilizado como tal por cualquier otra aplicación que lo necesite.

Partiendo de una imagen binaria, por tanto con sólo dos niveles de gris, el reconocimiento se realizará comparando unos patrones o plantillas que contiene todos los posibles caracteres.

Existen diversos problemas a la hora de llevar a cabo las funciones de reconocimiento de caracteres:

- Los niveles de grises introducidos no pertenecen a la imagen original.
- La resolución de las imágenes, ya que puede introducir ruido en ellas.
- La distancia de separación entre los caracteres puede ser una dificultad.
- Acoplamiento de dos o más caracteres por píxeles que pueden producir algún error.

Los pasos que lleva a cabo el algoritmo de OCR de la librería utilizada se basa en las siguientes cuatro etapas:

1. Binarización.
2. Fragmentación o segmentación de la imagen.
3. Adelgazamiento de los componentes.
4. Comparación con patrones.

El uso de esta librería es imprescindible ya que nos evita tener que diseñar un algoritmo específico propio para el reconocimiento de los caracteres del contador. Esta librería específica de OCR encontrada para el lenguaje de JAVA es una versión actualmente en prueba.

5.4 Herramientas de trabajo

El uso de un lenguaje como JAVA hace que todo lo que rodea el desarrollo de una aplicación basada en él sea relativamente sencillo. El uso extendido a nivel mundial de este lenguaje nos proporciona un gran abanico de herramientas que aportan al desarrollador un amplio abanico de posibilidades, haciendo que el trabajo resulte al final fácil y flexible.

Como hemos visto en el anterior apartado, se decidió elegir como entorno de desarrollo Eclipse. Al no tratarse de un desarrollo software individual, hemos creído conveniente y necesario la utilización de un sistema de control de versiones software, comúnmente conocido como Software Version Number (SVN). Aunque podía haberse realizado de forma manual, disponer de herramientas que facilitan esta gestión supone una gran comodidad a la hora de trabajar de manera independiente ya que la sincronización hace que el avance sea ininterrumpido y dinámico.

El desarrollo documental se ha realizado bajo el control de Google Docs, (2015). Al igual que ocurría con la parte de trabajo software, la necesidad de trabajar de una manera independiente hacían que fuera necesario un entorno donde se dispone de accesibilidad y actualización en cualquier momento. También resulta de gran utilidad a la hora de corregir errores o compartir la diferente información de cada miembro del proyecto, así la información se almacena en servidores de Internet. Para la realización de las distintas figuras así como diagramas de clases o casos de uso, utilizamos una aplicación de Google Drive llamada Draw.io (2015).

Además de los repositorios, se utilizó tanto el correo electrónico Gmail (2015) como Dropbox (2015) para intercambios puntuales de otros archivos que fueron necesarios para el desarrollo del proyecto, como por ejemplo librerías o imágenes del propio contador de gas.

5.5 Problemas y resolución

Este proyecto resultó para nosotros un campo totalmente nuevo como es el tratamiento de imágenes así como el de reconocimiento de caracteres. El desarrollo de una aplicación para tal fin ha supuesto un importante trabajo de investigación, líneas y posibilidades. Esto ha hecho que en ocasiones, el trabajo no se finalizara de forma satisfactoria, haciendo que o bien se desecharan o se buscaran soluciones alternativas al respecto.

Al no ser el lenguaje utilizado específico para el tratamiento de imágenes, uno de los principales problemas fue el encontrar una librería adaptada a JAVA para el tratamiento de imágenes, ya que se requería la implementación de muchos métodos teniendo la necesidad de acceder con precisión a todas las zonas de nuestras imágenes.

El hecho de no trabajar con imágenes de alta calidad ni definición hizo que fuera francamente difícil localizar con exactitud la zona del consumo del contador ya que para ello, nos basamos en buscar la parte roja. El método propuesto encontraba zonas rojas que no se correspondían con la de interés real. Finalmente, seleccionando la constante como un parámetro más preciso se pudo mejorar el algoritmo para que sólo seleccionara los píxeles de la tonalidad deseada.

Por último, para la parte de reconocimiento, se utilizó la mencionada librería ASPRISE, que resultó adecuada para el número de referencia del contador. Sin embargo, los caracteres del consumo poseen un formato inusual, además de la posibilidad de que no estén bien situados debido al movimiento de las ruedas giratorias que producen el cambio a medida que el consumo aumenta. Estas circunstancias han hecho que finalmente el reconocimiento haya sido complicado,

no habiéndose logrado totalmente la resolución del problema, quedando éste pendiente para trabajos de investigación específicos orientados a tal fin.

CAPÍTULO 6: Fase de pruebas

El objetivo de la realización de pruebas respecto al desarrollo de la aplicación es cerciorarnos y comprobar el correcto funcionamiento del sistema, así como garantizar que cumple con todos los requisitos propuestos en la fase de análisis.

Se han realizado una serie de pruebas que se enumeran a continuación:

1. Carga de imágenes.
2. Escala Grises.
3. Localización de las zonas de interés.
4. Binarización.
5. Lectura de los caracteres específicos.

6.1 Prueba 1: Carga de imágenes

El objetivo de esta primera prueba es exactamente verificar el correcto funcionamiento de la aplicación al abrir una determinada imagen. Se debe poder mostrar imágenes de tipo "jpg", "png" y "bmp".

Además se realiza un re-escalado a la imagen utilizando propiedades por defecto de la librería al utilizar un objeto de tipo "*BufferedImage*" de JAVA para poder mostrar dicha imagen de manera correcta visualmente, a la vez que se ayuda a simplificar y disminuir el coste computacional de los métodos y algoritmos al trabajar siempre con la misma escala.

La figura 17 muestra una imagen original cuyas dimensiones son de 2592x1944 píxeles (ancho x alto). Esta misma imagen se muestra en la figura 18,

la cual queda correctamente situada en el panel de interfaz a la vez que redimensionada al tamaño de 600x450 píxeles.

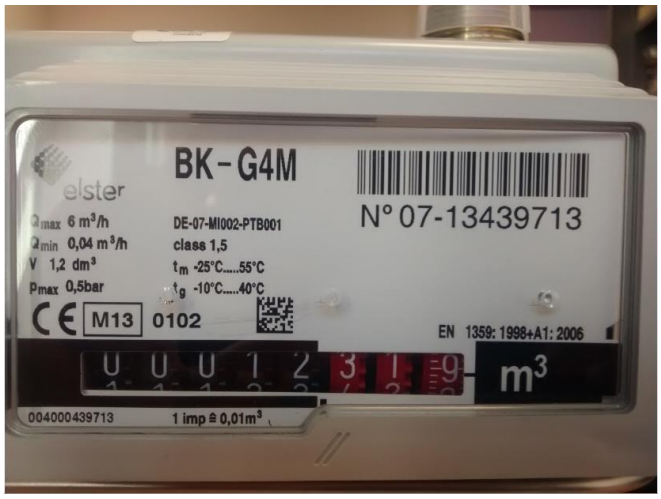


Figura 16. Imagen contador tamaño 2592x1944.

Figura 17. Imagen contador redimensionada a 600x450.

6.2 Prueba 2: Localización de las zonas de interés

Lo que se pretende con esta prueba es comprobar que independientemente de la imagen dada, la aplicación es capaz de localizar las partes de interés que contienen el número de referencia del contador y su consumo.

Llegamos a la conclusión que la manera más efectiva sería a partir de conseguir encontrar la parte roja del contador ya que, a nuestro parecer, hace que la búsqueda del número de referencia del contador así como el consumo resulte más simple puesto que conocemos la distancia a la que se encuentran los caracteres en ambas zonas de interés. Como se explicó más detalladamente en el capítulo 2, el algoritmo consiste en hacer un recorrido por toda la matriz de la imagen e ir marcando los píxeles rojos. Una vez localizados y al haber realizado la redimensión de la imagen, conocemos los parámetros necesarios para encuadrar las zonas de interés sobre los que se lleva a cabo el reconocimiento de los caracteres. En la figura 18 se muestra una imagen original a la izquierda, junto con el resultado de la identificación de la zona roja, así como los recuadros que señalan las zonas de interés donde se ubican las secuencias de caracteres a reconocer.

Figura 18. Imagen original (izquierda), con localización de las zonas de interés (derecha).

6.3 Prueba 3: Escala grises

Lo que se pretende con esta segunda prueba es verificar la obtención de la imagen a tratar en escala grises.

Partiendo de la imagen original que podemos ver en la figura 6.2 (izquierda) obtendremos la imagen en escala de grises. Para ello recorremos la matriz de píxeles de la imagen convirtiendo cada píxel realizando las operaciones necesarias para poder conseguir la imagen deseada. Dichas operaciones consisten en la transformación de la imagen en el modelo de color RGB al modelo HSI, quedándonos con la componente de intensidad, que es realmente la de grises. El resultado obtenido es el mostrado en la figura 19 (derecha), que permite comparar la imagen original con la imagen procesada.

Figura 19. Imagen original (izquierda), imagen en escala de grises (derecha).

6.4 Prueba 4: Binarización

Tras la localización de las zonas de interés y la transformación desde el modelo de color para obtener la imagen de grises, se realiza la segmentación de la imagen mediante umbralización por Otsu para obtener una imagen binaria que contiene información de partes relevantes. La zona de consumo aparece con sus píxeles negros, si bien con los caracteres en blanco. La zona de identificación del usuario contiene sus píxeles, caracteres y código de barras, en negro sobre fondo blanco. La figura 20 muestra una imagen original (izquierda) y la correspondiente imagen binarizada mediante Otsu (derecha).

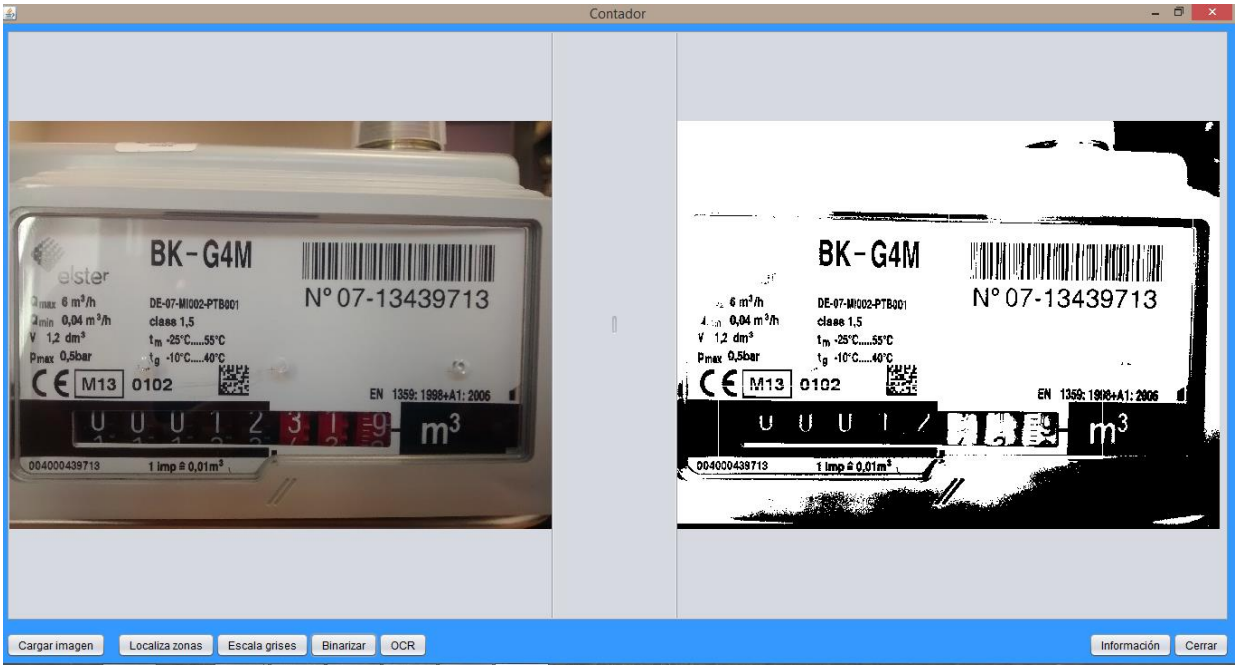


Figura 20. Imagen original, imagen binarizada mediante Otsu.

6.5 Prueba 5: Lectura de caracteres específicos

Con esta prueba, lo que se pretende es comprobar que una vez localizadas las zonas y con la imagen binarizada es posible reconocer los caracteres de interés. Como se explicó previamente, este reconocimiento tiene su fundamento en la librería OCR ASPRISE, con lo cual, se verifica que la aplicación llega a limpiar la imagen de ruido subyacente con el fin de reconocer los caracteres con la mayor precisión posible. En la figura 21 se muestra una imagen original (izquierda), apareciendo el número de referencia en la parte derecha, indicando que el reconocimiento ha sido satisfactorio.

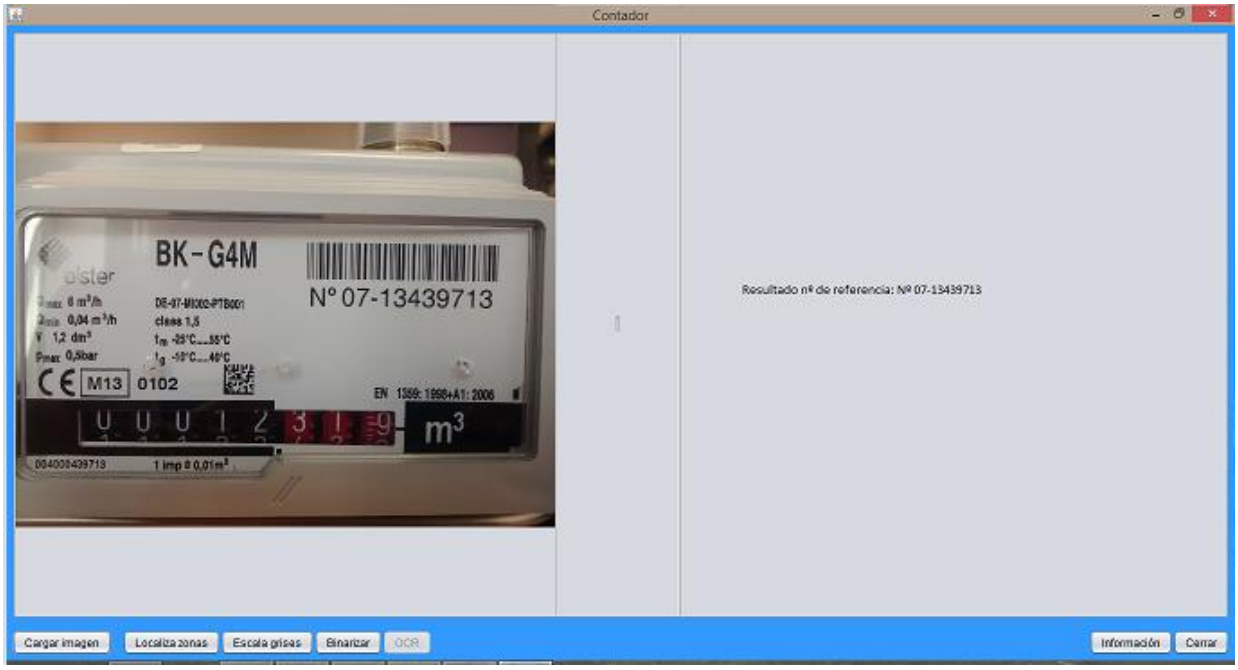


Figura 21. Imagen original (izquierda), resultado de la lectura de caracteres relativos al código del contador (derecha).

Como se muestra en la figura, sólo se muestra con precisión el número de referencia del contador. Surge un grave problema a la hora de reconocer el consumo. Debido a que estos caracteres son muy específicos y por lo tanto poco comunes no se puede obtener un resultado mínimamente preciso. Además, el hecho de que los caracteres sean blancos sobre negro dificulta aún más el trabajo de lectura.

CAPÍTULO 7: Conclusiones

El proyecto desarrollado ha tenido su motivación en un interés empresarial real, consistente en la automatización del proceso de lectura de contadores de gas, con el fin de automatizar un proceso tedioso y complicado, que ha de redundar necesariamente en una mayor efectividad empresarial a la vez que se reducen los costes derivados del proceso de lectura mediante el procedimiento clásico.

El proyecto realizado ha seguido el ciclo de vida típico de cualquier aplicación que se puede desarrollar a nivel educativo o empresarial, intentando poner en práctica técnicas de ingeniería del software y afianzando distintos conceptos aprendidos durante la carrera tales como el uso de patrones (MVC) o documentación utilizando diagramas de clases.

Nuevos conceptos han sido aplicados en este proyecto y se ha llegado a ampliar desarrollos relativos al tratamiento y manipulación de imágenes. Para ello, se han utilizado diferentes librerías software y se ha mejorado considerablemente el planteamiento mediante el lenguaje de programación, en nuestro caso, JAVA.

En cuanto al diseño de la interfaz gráfica se ha conseguido una estructura eficaz, capaz de proporcionar al usuario un punto de vista amigable y eficiente mediante el uso fácil de la misma.

La implementación de los distintos algoritmos, tales como los tipos de binarizaciones o la localización de rojos, ha permitido la adaptación de nuevos conceptos y técnicas utilizadas en el tratamiento de imágenes aplicado al

reconocimiento óptico de caracteres en imágenes digitales, abriendo todo un campo nuevo de posibilidades en el que todavía queda mucho por hacer e investigar.

Con respecto a la fiabilidad en el reconocimiento del número de referencia, éste se detecta en aproximadamente un 85% de las veces en los contadores utilizados. En los casos de error se ha identificado que el problema se localiza en el proceso de captura de la imagen ya sea por falta o exceso de luz, lejanía, o la luz de flash que hace que el cristal del contador la refleje, haciendo prácticamente imposible el reconocimiento, incluso para el ojo humano. Tal y como se ha mencionado previamente, estos mismos problemas aparecen en el reconocimiento de los caracteres correspondientes al consumo agravados aún más por los problemas anteriormente mencionados. En este caso el reconocimiento no ha sido suficientemente satisfactorio.

Como conclusión final cabe destacar el compromiso de los dos miembros de este proyecto a la hora de conseguir los diferentes objetivos y plazos previstos no sólo por los responsables de dirigir el trabajo, sino también por los propios componentes, haciendo que haya resultado ser un trabajo en el que también se ha aprendido a trabajar en equipo de una manera organizada y responsable de cara a la vida laboral.

CAPÍTULO 8: Trabajo futuro

Mediante el desarrollo realizado se ha conseguido una aplicación con los resultados indicados. En este sentido conviene señalar que la misma es todavía mejorable, necesitando nuevos desarrollos y planteamientos para dar solución a la problemática que se quiere resolver. La modularidad de la arquitectura del sistema propuesta facilitará este proceso en futuros desarrollos, ya que permite la incorporación de nuevos módulos y funcionalidades sin afectar al resto de módulos..

Una metodología que se puede considerar para mejorar la eficiencia de la aplicación es el uso redes neuronales como reconocedor de caracteres. Aunque su uso supondría elevar la complejidad de implementación, que jugaría en su contra, el reconocimiento de los caracteres sería mucho más fiable, ya que las redes neuronales tienen la capacidad de aprender de sus errores, de forma que con un buen entrenamiento se logra una gran eficiencia a la hora de clasificar caracteres.

Es necesario tener más en cuenta las situaciones relativas a la captura de imágenes, en referencia a las condiciones de visibilidad adversas. La lejanía del contador con respecto al dispositivo de captura de la imagen, el flash u otros reflejos, ruido de la imagen y muchas otras condiciones ambientales hacen que el reconocimiento de los números sea en ocasiones muy difícil de lograr. Existen en la literatura proyectos que tratan de salvar estas dificultades con algoritmos más sofisticados como el propuesto por Bunke y Wang (1997). La robustez de los métodos se incrementaría incorporando nuevas funcionalidades derivadas de esta investigación.

En cuanto al número de consumo no hemos sido capaces de reconocerlo con una mínima fiabilidad. Se debería realizar un estudio previo de los caracteres que se utilizan, considerando aspectos tales como su tamaño, forma, inclusión en su entorno. Esto llevaría a plantear el desarrollo de una nueva librería de reconocimiento para este tipo de caracteres y su disposición en el contador.

CAPÍTULO 9: Bibliografía

- Arlow, J., y Neustadt, I. (2006). "UML 2". ANAYA, 2006.
- ASPRISE Java OCR (2015). http://asprise.com/java-ocr-api-overview.html (accedido Junio 2015)
- Bostjan (2015). Java Image Binarization using Otsu's algorithm http://developer.bostjan-cigan.com/java-image-binarization/ (accedido Junio 2015)
- Bunke, H. y Wang, P.S.P. (Eds.) (1997). Handbook of Character Recognition and Document Image Analysis. World Scientific.
- Draw.io (2015). www.draw.io/ (accedido Junio 2015)
- Dropbox (2015) www.dropbox.com/ (accedido Junio 2015)
- Gmail (2015) https://www.gmail.com/ (accedido Junio 2015)
- GoogleDocs (2015) https://docs.google.com/ (accedido Junio 2015)
- Laganière, R. (2011) OpenCV 2 Computer Vision Application Programming Cookbook, Packt Publishing, Olton, UK.
- Madrileña Red de Gas (2015) http://www.madrilena.es/ (accedido Junio 2015)
- Otsu, N. (1979). A threshold selection method from gray-level histogram. IEEE Transactions on System Man and Cybernetics 9, 62-66, 1979.
- Pajares, G. y de la Cruz, J.M (2007). Visión por computador. Imágenes digitales y aplicaciones. Ra-Ma, Madrid.
- Software Version Number (SVN) (accedido Junio 2015). https://wiki.eclipse.org/Eclipse_Project_Update_Sites

ANEXO: Manual de usuario

La aplicación está formada por una interfaz gráfica principal la cual permitirá al usuario ejecutar y realizar las funcionalidades de la aplicación. Se muestra a continuación en la figura A.1, la ventana principal de acceso a la misma.

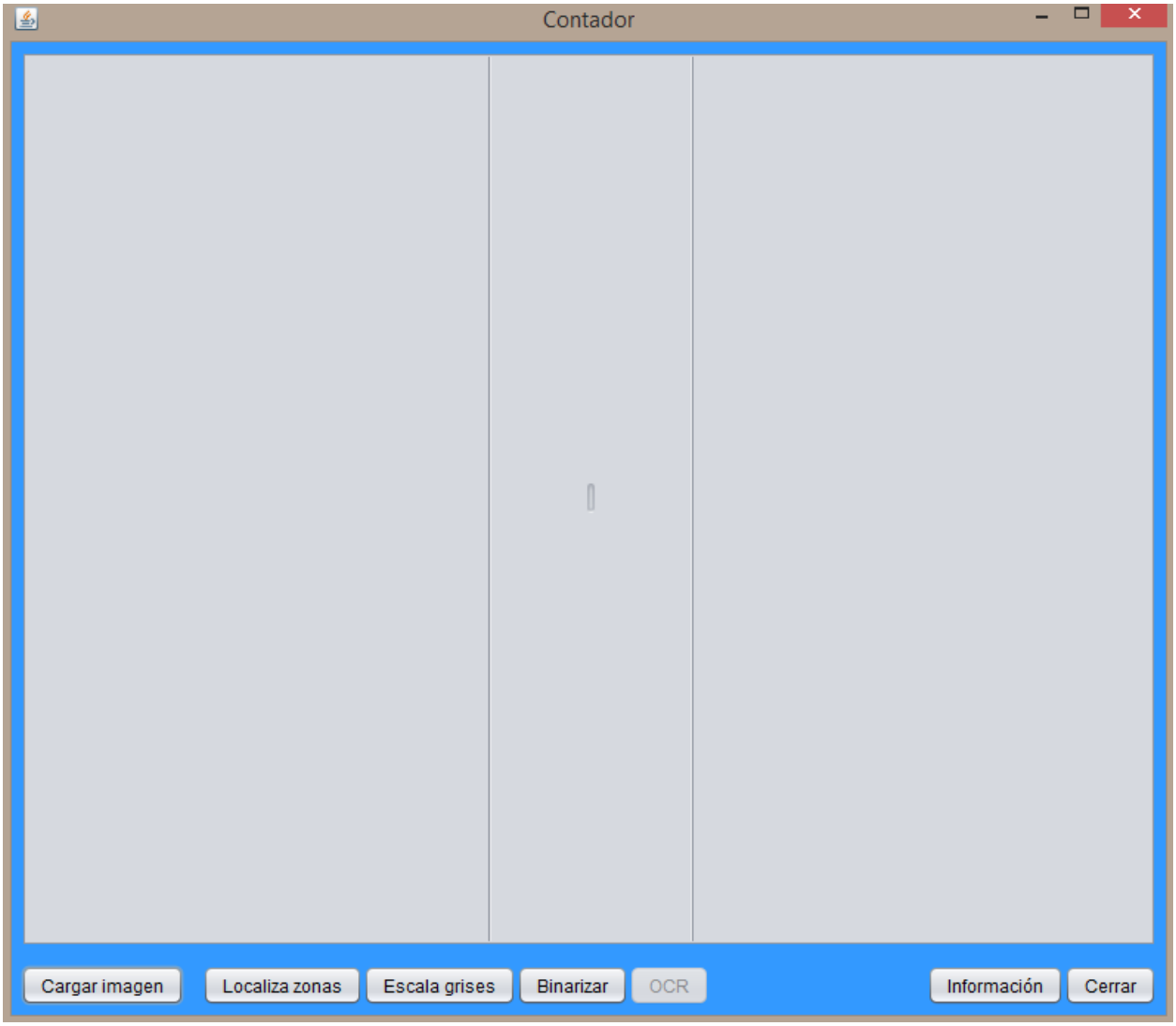


Figura A.1. Ventana inicial de la aplicación.

Esta ventana inicial muestra en la parte inferior los siguientes botones:

CARGAR IMAGEN

Antes de poder aplicar cualquier función de binarización o los algoritmos de localización a una determinada imagen, es necesario cargarla. Para ello, una vez pulsado el botón se abrirá automáticamente un cuadro de diálogo para seleccionar la ruta de la imagen que queremos abrir para su posterior tratamiento, tal como se muestra en la figura A.2.

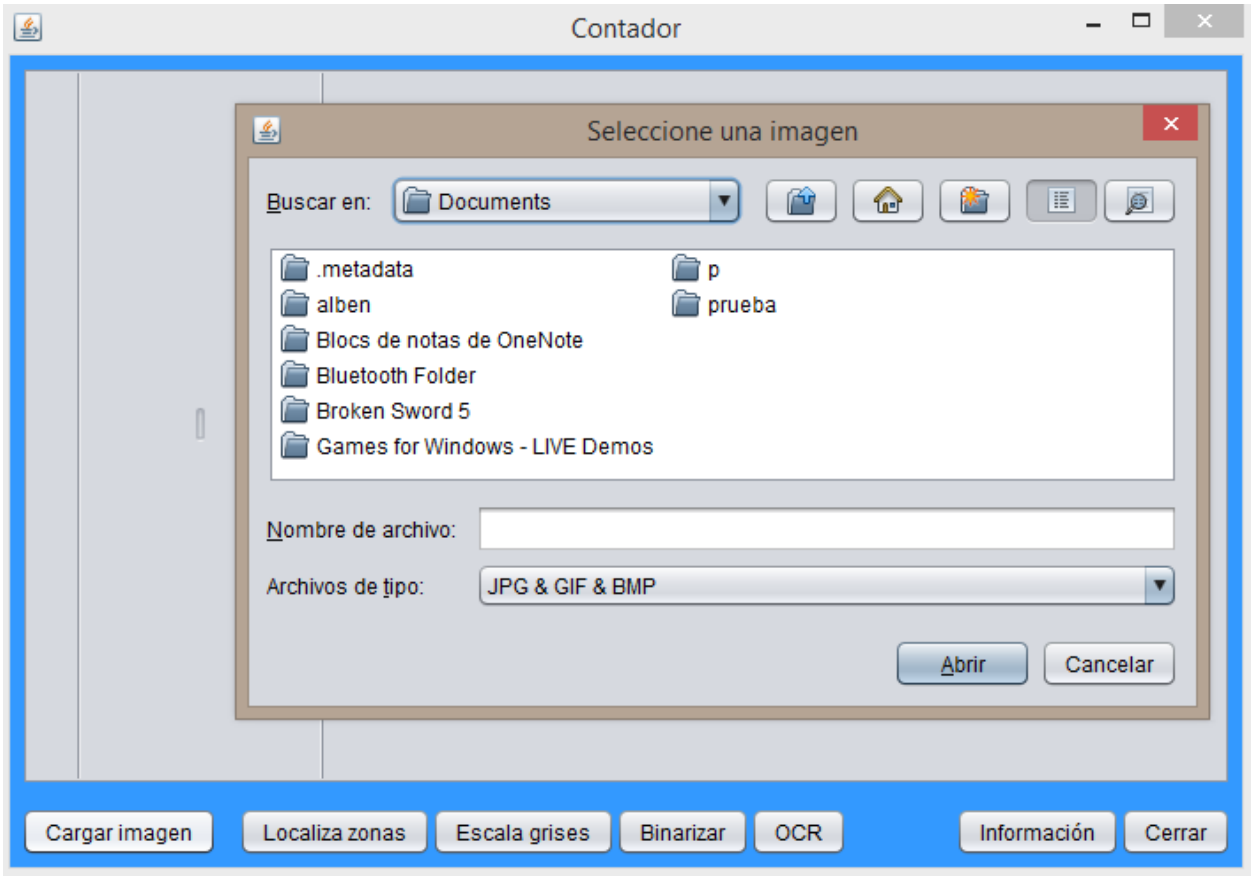


Figura A.2. Selección de imagen a tratar.

Esta ventana nos permite seleccionar la imagen y abrirla en la aplicación. Se debe pulsar el botón *Abrir*, o si por el contrario deseamos volver a la ventana de inicio, bastará con pulsar el botón *Cancelar.*

Figura A.3. Imagen cargada del contador de gas.

Tras elegir la ruta donde se encuentra ubicada la imagen y pulsar *Abrir*, si anteriormente se había cargado otra imagen, se sustituirá por la elegida actualmente. Sin embargo, tanto la imagen principal como las imágenes de las zonas que se habían localizado seguirán guardadas, ya que como vimos en capítulos anteriores se guardaban al aplicar el algoritmo de reconocimiento óptico de caracteres para mayor comodidad si se quisiera comprobar el resultado del algoritmo.

LOCALIZA ZONAS

Figura A.4. Localización de zonas.

Pulsando este botón se procederá a aplicar el algoritmo de localización. Aunque no sea una imagen precisa, la aplicación está diseñada para que siempre muestre las zonas en las que deberían estar el número de referencia y el consumo basándose en los píxeles rojos de la imagen. Además, para que el botón funcione correctamente, debe haber una imagen cargada, en caso contrario se mostrará el siguiente mensaje:

Figura A.5. Mensaje de aviso de carga imagen.

ESCALA GRISES

Este botón no es estrictamente necesario ya que el paso a la escala de grises se podría haber realizado cuando se pulsa el botón de binarización. Sin embargo, para una mayor comprensión de la aplicación hemos creído conveniente ver cómo se va tratando la imagen paso a paso hasta llegar al resultado final.

Figura A.6. Escala Grises.

BINARIZACIÓN

Las imágenes binarias siempre se obtienen a partir de imágenes de niveles de gris. Si se ha realizado un buen escalado a grises con un umbral adecuado la pérdida de información será mínima y resultará mucho más fácil para que el algoritmo OCR realice el reconocimiento de los caracteres.

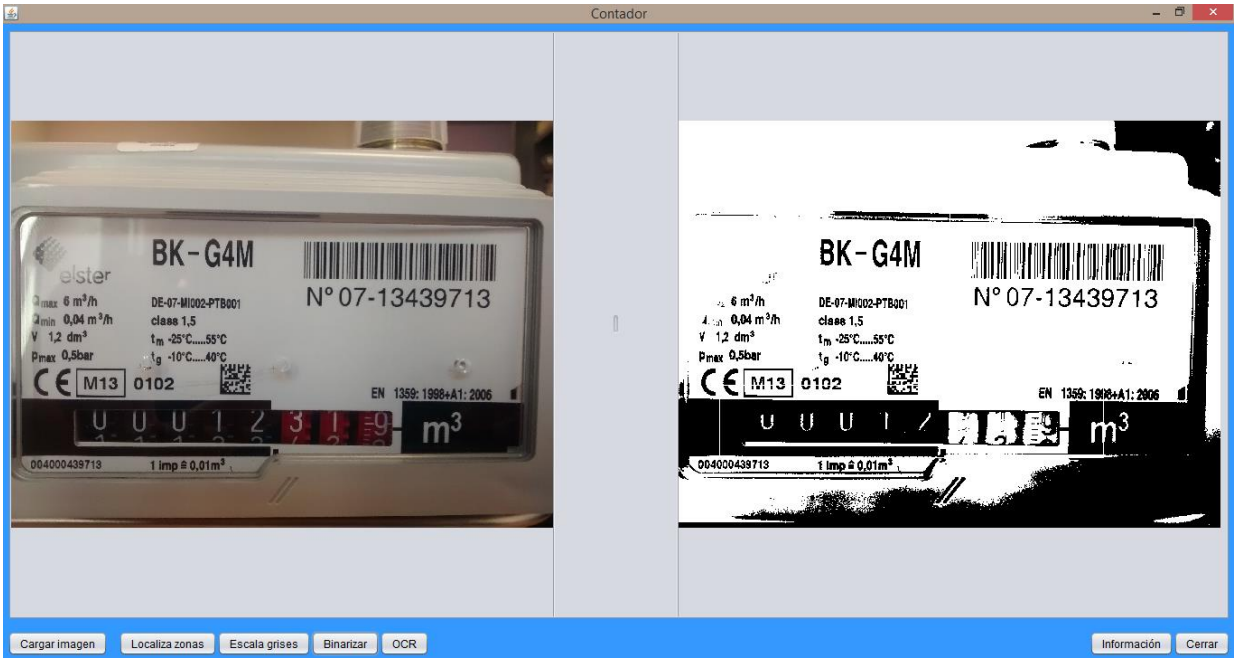


Figura A.7. Binarización.

OCR

Si está habilitado significa que existe una imagen cargada y que se ha estudiado previamente pulsando los otros botones. Por lo tanto, se puede proceder a la lectura apareciendo el resultado por pantalla o un mensaje de error en caso de que haya fallado la lectura.

Figura A.8. OCR.

INFORMACIÓN

Este botón simplemente proporciona información sobre los autores del proyecto y el año en el que se ha desarrollado la aplicación, la figura A.4 muestra el mensaje de información correspondiente.

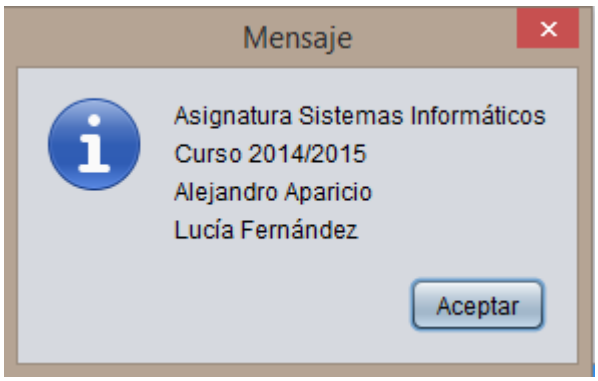


Figura A.9. Cuadro de diálogo sobre nosotros.

CERRAR

Se finaliza el proceso de identificación cerrando la aplicación.