ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA (ICAI)

GRADO EN INGENIERÍA ELECTROMECÁNICA

ESPECIALIDAD ELECTRÓNICA

VISIÓN ARTIFICIAL DE DOCUMENTOS

Autor: José Soria Soto

Director: Juan Antonio Talavera Martín

Madrid

Julio 2018

AUTORIZACIÓN PARA LA DIGITALIZACIÓN, DEPÓSITO Y DIVULGACIÓN EN RED DE PROYECTOS FIN DE GRADO, FIN DE MÁSTER, TESINAS O MEMORIAS DE BACHILLERATO

1º. Declaración de la autoría y acreditación de la misma.

El autor **D. JOSÉ SORIA SOTO**

DECLARA ser el titular de los derechos de propiedad intelectual de la obra: **VISIÓN ARTIFICIAL DE DOCUMENTOS**, que ésta es una obra original, y que ostenta la condición de autor en el sentido que otorga la Ley de Propiedad Intelectual.

2º. Objeto y fines de la cesión.

Con el fin de dar la máxima difusión a la obra citada a través del Repositorio institucional de la Universidad, el autor **CEDE** a la Universidad Pontificia Comillas, de forma gratuita y no exclusiva, por el máximo plazo legal y con ámbito universal, los derechos de digitalización, de archivo, de reproducción, de distribución y de comunicación pública, incluido el derecho de puesta a disposición electrónica, tal y como se describen en la Ley de Propiedad Intelectual. El derecho de transformación se cede a los únicos efectos de lo dispuesto en la letra a) del apartado siguiente.

3º. Condiciones de la cesión y acceso

Sin perjuicio de la titularidad de la obra, que sigue correspondiendo a su autor, la cesión de derechos contemplada en esta licencia habilita para:

a) Transformarla con el fin de adaptarla a cualquier tecnología que permita incorporarla a internet y hacerla accesible; incorporar metadatos para realizar el registro de la obra e incorporar "marcas de agua" o cualquier otro sistema de seguridad o de protección.
b) Reproducirla en un soporte digital para su incorporación a una base de datos electrónica, incluyendo el derecho de reproducir y almacenar la obra en servidores, a los efectos de garantizar su seguridad, conservación y preservar el formato.
c) Comunicarla, por defecto, a través de un archivo institucional abierto, accesible de modo libre y gratuito a través de internet.
d) Cualquier otra forma de acceso (restringido, embargado, cerrado) deberá solicitarse expresamente y obedecer a causas justificadas.
e) Asignar por defecto a estos trabajos una licencia Creative Commons.
f) Asignar por defecto a estos trabajos un HANDLE (URL *persistente).*

4º. Derechos del autor.

El autor, en tanto que titular de una obra tiene derecho a:

a) Que la Universidad identifique claramente su nombre como autor de la misma
b) Comunicar y dar publicidad a la obra en la versión que ceda y en otras posteriores a través de cualquier medio.
c) Solicitar la retirada de la obra del repositorio por causa justificada.
d) Recibir notificación fehaciente de cualquier reclamación que puedan formular terceras personas en relación con la obra y, en particular, de reclamaciones relativas a los derechos de propiedad intelectual sobre ella.

5º. Deberes del autor.

El autor se compromete a:

a) Garantizar que el compromiso que adquiere mediante el presente escrito no infringe ningún derecho de terceros, ya sean de propiedad industrial, intelectual o cualquier otro.
b) Garantizar que el contenido de las obras no atenta contra los derechos al honor, a la intimidad y a la imagen de terceros.
c) Asumir toda reclamación o responsabilidad, incluyendo las indemnizaciones por daños, que pudieran ejercitarse contra la Universidad por terceros que vieran infringidos sus derechos e intereses a causa de la cesión.

d) Asumir la responsabilidad en el caso de que las instituciones fueran condenadas por infracción de derechos derivada de las obras objeto de la cesión.

6º. Fines y funcionamiento del Repositorio Institucional.

La obra se pondrá a disposición de los usuarios para que hagan de ella un uso justo y respetuoso con los derechos del autor, según lo permitido por la legislación aplicable, y con fines de estudio, investigación, o cualquier otro fin lícito. Con dicha finalidad, la Universidad asume los siguientes deberes y se reserva las siguientes facultades:

- La Universidad informará a los usuarios del archivo sobre los usos permitidos, y no garantiza ni asume responsabilidad alguna por otras formas en que los usuarios hagan un uso posterior de las obras no conforme con la legislación vigente. El uso posterior, más allá de la copia privada, requerirá que se cite la fuente y se reconozca la autoría, que no se obtenga beneficio comercial, y que no se realicen obras derivadas.
- La Universidad no revisará el contenido de las obras, que en todo caso permanecerá bajo la responsabilidad exclusive del autor y no estará obligada a ejercitar acciones legales en nombre del autor en el supuesto de infracciones a derechos de propiedad intelectual derivados del depósito y archivo de las obras. El autor renuncia a cualquier reclamación frente a la Universidad por las formas no ajustadas a la legislación vigente en que los usuarios hagan uso de las obras.
- La Universidad adoptará las medidas necesarias para la preservación de la obra en un futuro.
- La Universidad se reserva la facultad de retirar la obra, previa notificación al autor, en supuestos suficientemente justificados, o en caso de reclamaciones de terceros.

Madrid, a 4 de JULIO de 2018

ACEPTA

Fdo. 

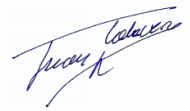
Motivos para solicitar el acceso restringido, cerrado o embargado del trabajo en el Repositorio Institucional:

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título **VISIÓN ARTIFICIAL DE DOCUMENTOS** en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el curso académico **2017/2018** es de mi autoría, original e inédito y no ha sido presentado con anterioridad a otros efectos. El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido tomada de otros documentos está debidamente referenciada.

Fdo.: JOSÉ SORIA SOTO Fecha: 3/ 7/ 2018

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO



Fdo.: JUAN ANTONIO TALAVERA MARTÍN Fecha: 4/ 7/ 2018

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA (ICAI)

GRADO EN INGENIERÍA ELECTROMECÁNICA

ESPECIALIDAD ELECTRÓNICA

VISIÓN ARTIFICIAL DE DOCUMENTOS

Autor: José Soria Soto

Director: Juan Antonio Talavera Martín

Madrid

Julio 2018

VISIÓN ARTIFICIAL DE DOCUMENTOS

Autor: **Soria Soto, José**

Director: Talavera Martín, Juan Antonio

Entidad colaboradora: Universidad Pontificia de Comillas, ICAI

RESUMEN DEL PROYECTO

1. INTRODUCCIÓN

El proyecto desarrollado en este documento trata sobre la elaboración de un programa de reconocimiento de texto en imágenes, para la identificación de información de interés en un documento que se presenta en forma de imagen.

Se aplican sobre dichas imágenes un proceso de digitalización de textos para el reconocimiento de la información, mediante el proceso de reconocimiento óptico de caracteres (OCR). Todo ello se realiza aplicando las técnicas de visión artificial pertinentes.

El desarrollo de dicho proceso de identificación de texto en imágenes se complementa con un lector de códigos de barra tipo PDF417. Esto se debe a que la información del documento a reconocer, además de en la imagen a procesar, se encuentra también encriptada en códigos de barras del tipo mencionado. Obtener la información de dichos códigos sirve para perfeccionar el reconocimiento final.

El proyecto se realiza utilizando C++ como lenguaje de programación y utilizando una estructura modular para facilitar su entendimiento y uso.

La idea final de este proyecto es la implantación de la aplicación desarrollada en cualquier entorno laboral, con el fin tanto de identificar la información existente en el documento como la posterior clasificación de este.

Todo el proceso se realiza con la pretensión de obtener los mejores resultados posibles, teniendo en cuenta en todo momento la eficiencia y eficacia de las soluciones desarrolladas.

2. METODOLOGÍA

En primer lugar para la realización de este proyecto se procede con un estudio previo de las tecnologías existentes tanto de motores OCR ya perfectamente desarrollados como de las herramientas y recursos necesarios para el desarrollo del programa. Tras este proceso de análisis, se toma la decisión de que el entorno de desarrollo integrado (IDE) utilizado sea Visual Studio 2013, que la librería de visión artificial sea OpenCV, y para el reconocimiento de los códigos de barras se utilizarán las librerías que hay implementadas en la aplicación de Dynamsoft Barcode Reader.

Para la realización de este programa se parte de imágenes externas, provenientes de un escáner o cámara, las cuales se integran en el proyecto ya como un archivo con formato de imagen.

El entendimiento de las técnicas de visión artificial, hace que las primeras fases del proyecto sean realizando pruebas en las que se aplican algunas de dichas técnicas para conocer su funcionamiento. Un primer ejemplo es el cargado de imágenes.

Posteriormente se pasa al tratamiento de imágenes. En una etapa de pre-procesamiento se aplican sobre la imagen a reconocer, técnicas de visión artificial para la mejor identificación de la información. Algunas de esas técnicas son la conversión a escala de grises, o la elección del valor umbral para la binarización de la imagen. Tras esto se llega a la fase de tratamiento de imágenes, en la cual se aplican las operaciones morfológicas necesarias para dejar la imagen, de la mejor manera posible, preparada para del reconocimiento. Sobre dicha imagen se encuentran los contornos que pueden ser posibles caracteres y se intentan reconocer.

En cuanto al reconocimiento, se elige el método de los k vecinos más próximos (KNN), el cual es fácil de implementar y proporciona resultados razonablemente buenos. Puesto que se elige este algoritmo, paralelamente al desarrollo del programa principal, se realiza otro programa de entrenamiento, en el cual a partir de una imagen con los caracteres de interés (alfabeto, números y principales signos de escritura) y tras pre-procesarla y

procesarla, se enseña manualmente a identificar la imagen de cada una de esas caracteres con su correspondiente valor en el código ASCII.

Tras el entrenamiento y el procesamiento de la imagen original, se realiza el reconocimiento. El hecho de que los resultados obtenidos no tengan la eficiencia deseada, es lo que lleva a usar los códigos PDF417 como complemento al reconocimiento.

Esta necesidad hace que se realice un programa de reconocimiento de códigos de barras. Para ello se utilizan las librerías previamente mencionadas.

Además de todo esto se irán perfeccionando en todo momento el programa en busca de mayor funcionalidad. Entre otras competencias, se introduce comunicación con el usuario para que se ejecute de una manera u otra el programa.

Finalmente se integran todas las fases desarrollado en un proyecto final que lo recoja todo, que esté bien estructurado y sea fácilmente entendible.

3. RESULTADOS

Los resultados, en términos globales, del proyecto se pueden considerar exitosos, ya que se cumplen los objetivos marcados de manera satisfactoria.

En cuanto a los resultados de las distintas fases de desarrollo del proyecto, también se consideran apropiados. Es cierto que se encuentran algunas complicaciones, como pueden ser la falta de eficacia a la hora del reconocimiento de texto o la ausencia de una interfaz de usuario, pero por otro lado, lo bueno es que se han conseguido suplir esas dificultades con otras técnicas o procedimientos.

Además de los objetivos referidos al desarrollo del programa, se consiguen los objetivos de aprendizaje, a lo largo de todo el proceso, relacionados con diversos ámbitos como, por ejemplo, la programación con técnicas de visión artificial y el tratamiento de imágenes entre otros.

4. CONCLUSIÓN

Se puede decir finalmente que este proyecto tiene una utilidad real, avalado por su funcionalidad y eficacia.

Se busca el realismo del proyecto y para ello además de los resultados útiles, se explican y comentan posibles casos prácticos de eventos que podrían darse como, la pérdida de información en los códigos de barras o la disposición de la posible información a reconocer. Todo ello, así como la disponibilidad de la información en varios formatos y ser capaz de su reconocimiento en ambos, le dan robustez y eficiencia al programa.

En cuanto a futuros desarrollos del trabajo se abren varios caminos. Entre ellos se destacan dos posibles. El primero es el perfeccionamiento de este proyecto, solventando las dificultades comentadas, por ejemplo implementando una interfaz propia. Puesto que hoy en día todo lo relacionado con el tema de “Machine Learning” está siendo estudiado e implementado en todos nuestros entornos, un segundo camino, más complicado, sería dotar de inteligencia al programa.

COMPUTER VISION FOR DOCUMENTS

Author:	**Soria Soto, José**
Director:	Talavera Martín, Juan Antonio
Collaborating organization:	Universidad Pontificia de Comillas, ICAI

ABSTRACT

1. INTRODUCTION

This text covers the development of a text recognition software. This software allows the user to identify useful information from a document with image format.

The images are subjected to a text digitalization process for information gathering through optical character recognition (OCR). The whole process takes place via pertinent artificial vision techniques.

This text identifying process based on images is complemented by a barcode reader type PDF417. That is because the information contained in the document to process is presented not only in image format, but also encrypted in barcode of the previous mentioned type. Gathering the information from those barcodes allows the final recognition to be more precise.

The project has been developed using C++ as programming language. In addition, it has been structured in modules to facilitate the use and understanding.

The main idea of this project is the implantation of the developed application in a working environment with the aim of identifying the information present in the document and the subsequent classification.

The whole process aims to obtain the best possible results, always taking into account the efficiency and effectiveness of the developed solutions.

2. METHODOLOGY

First, a preliminary study is performed of the existent technologies both on complete OCR engines and necessary assets for the development of the program. After this analysis, Visual Studio 2013 is determined as the most suitable option for the Integrated development environment (IDE). Also, the chosen artificial vision library is OpenCV and the used barcode libraries will be the already implemented in the Dynamsoft barcode reader software.

The origin of the external images used for the development is a scanner or a camera. Those images are integrated into the document already with image format.

The understanding of the artificial vision techniques results in the first phases of the project being mainly test performings. In those tests, some of these techniques are applied to know their operation. A first example is the loading of images.

Afterwards, the image is processed. In a pre-processing stage, artificial vision techniques are applied to the image to be recognized in order to improve the information identification. Some of these techniques include the conversion to gray scale, or the choice of the threshold value for the image binarization. After that, the image treatment phase is reached, in which the necessary morphological operations are applied to leave the image ready for recognition in the best possible way. On this image, the contours that may result in possible characters are located and a recognition is performed.

Regarding the recognition, the method of the nearest k neighbors (KNN) is chosen. This method is easy to implement and provides reasonably good results. Since this algorithm is chosen, parallel to the development of the main program, another training program is developed. This program takes an image with the characters of interest (alphabetical, numerical and main signs of writing) and, after pre-processing and processing it, it is manually taught to identify the image of each one of those characters with its corresponding value in the ASCII code.

After the training and processing of the original image, recognition is performed. However, the results obtained do not have the desired efficiency so it is necessary to use the PDF417 codes as a complement to the recognition. It is precisely this need that leads

to the development of a bar code recognition program in which the previously mentioned libraries are used.

In addition to all of this, the program will be refined at all times in search of greater functionality. Among other competences, communication with the user is introduced so the program is executed in many ways.

Finally, all the modules are integrated into a final project with a proper structure and easy to understand.

3. RESULTS

The overall results of the project can be considered successful, since they meet the objectives set in a satisfactory manner.

Regarding the results of the different phases of the project development, they are also considered appropriate. It is true that there are some complications, such as the lack of efficiency at the time of text recognition or the absence of a user interface. On the other hand, these difficulties have been overcome with other techniques or procedures.

Together with the objectives related to the development of the program, learning milestones are achieved throughout the process, related to various areas such as programming with artificial vision techniques or the treatment of images.

4. CONCLUSIONS

It can be finally said that this project has a real utility, backed by its functionality and effectiveness.

The realism of the project is sought. That is why, in addition to the useful results, possible cases of events that could occur are explained and discussed. Examples of this would be the loss of information in the bar codes or the provision of the possible information to be recognized. The results and the examples, together with the availability of information in various formats and their recognition, give the program robustness and efficiency.

Regarding future developments of this work, several paths open up. Among them, two possible ones stand out. The first one is the improvement of the current project, solving

the difficulties discussed such us the implementation of a user interface. Since today Machine Learning is on everyone's lips, a second and more complicated path would be to provide intelligence to the program.

Índice de la memoria

Índice de figuras

Índice de tablas

Parte I MEMORIA

Capítulo 1 INTRODUCCIÓN

1.1 INTRODUCCIÓN

Este proyecto trata sobre la realización de un programa o software de visión artificial, y por tanto de métodos de adquisición, procesamiento y análisis de imágenes, con el objetivo del reconocer documentos mediante la aplicación de técnicas de visión artificial desarrolladas para robots industriales.

Para ello se partirá de imágenes de documentos provenientes de cámaras o escáner y se trata de aplicar técnicas de visión artificial para identificar el contenido de la imagen mediante la digitalización de textos con el proceso de reconocimiento óptico de caracteres (OCR).

A partir de las imágenes de los distintos documentos a analizar, mediante el programa que se desarrolla en este proyecto, se identificará la información útil y el tipo de documento del que se trata para su posterior clasificación.

1.2 MOTIVACIÓN DEL PROYECTO

En la actualidad vivimos en un mundo cada vez más tecnológico en el que gracias a la curiosidad y al deseo de tener una vida más cómoda y feliz ha evolucionado de una manera tremendamente rápida siguiendo una progresión geométrica. Dentro de este universo en el que todo está siendo automatizado la visión artificial juega un papel fundamental.

Con el objetivo de hacer un mundo tecnológico y automatizado hay que permitir y ser capaz de que ordenadores, robots o cualquier tipo de máquina pueda realizar los trabajos que hace no tantos años tendría que realizar un ser humano, y así, además de hacer ese trabajo más eficiente, dotar al ser humano de mayor libertad para realizar otras tareas. Puesto que la idea es que esas máquinas realicen una tarea como podría hacerlo un ser humano no es extraño pensar que la visión artificial ha de jugar un papel primordial en esa máquina en la misma medida que el sentido de la vista en los humanos.

La visión es un sentido aún con un largo camino de investigación con el fin de poder entender como los seres vivos comprendemos la realidad a través de nuestros ojos. Debido a que este entendimiento es gracias a nociones tanto inertes como aprendidas de geometría, física o estadística entre otras muchas, se quiere dotar de ellas también a nuestros sistemas tecnológicos para que capten la realidad y actúen cómo nosotros.

Una de las causas que justifican la elección de este proyecto es debido a querer realizar un trabajo práctico en el que se pueda ver el funcionamiento real de una manera directa, poniendo en práctica los conocimientos aprendidos durante la carrera sobre todo de la parte de programación, a la vez que aprendo a usar otras instrucciones y disciplinas necesarias para poder realizar esta tarea de la mejor manera posible.

1.3 OBJETIVOS

En el caso de este proyecto cabe destacar que los objetivos a continuación mencionados, sirven para dar nombre posteriormente a los distintos apartados del capítulo de “Desarrollo y resultados”, donde se explica todo el desarrollo del proyecto.

- Estudiar, analizar y montar las aplicaciones y elementos necesarios para el desarrollo del proyecto.
    - Descarga del IDE y de las librerías de visión artificial y de códigos de barras.
    - Preparación del IDE e incorporación de las librerías de visión artificial y de códigos de barras.
- Pruebas aisladas de cargado de imágenes y pre-procesamiento.
    - Método de cargado de imágenes.
    - Pre-procesamiento.
- Pruebas asiladas de procesamiento de imágenes.
    - Preparación de la imagen para el reconocimiento de caracteres.
    - Método KNN. Entrenamiento y reconocimiento.
- Pruebas aisladas de procesamiento de códigos de barras
    - Códigos PDF417. Preparación y reconocimiento.
- Implementar primitivas del lenguaje para el cargado de imágenes, pre-procesamiento y lenguaje del usuario.
    - Lenguaje de usuario.
    - Cargado de imágenes.
    - Pre-procesamiento.
- Implementar primitivas de procesamiento y salida.
    - Procesamiento de imágenes.
    - Procesamiento de códigos de barras.
    - Primitivas de salida.
- Integración y pruebas.
    - Integración.
    - Pruebas de las distintas configuraciones.
- Optimización.

1.4 RECURSOS / HERRAMIENTAS EMPLEADAS

- Ordenador

  Necesario para todas las fases del proyecto puesto que es la única plataforma de trabajo en este proyecto al tratarse de un proyecto de programación. Cualquier modelo de ordenador suficientemente bueno es adecuado en principio para la realización del proyecto.

- Visual Studio

  Es el entorno de desarrollo integrado (IDE) que se utiliza. Es de fácil acceso y su simplicidad y la gran cantidad de funciones y servicios la hacen la opción elegida. Además, es un entorno que acepta gran cantidad de lenguajes de programación como C, C++, C#, Basic, Java, etc. Será utilizado como base donde escribir el código fuente.

- OpenCV

  Es la biblioteca de visión artificial elegida para este proyecto. Se trata de una biblioteca libre, desarrollada en C y C++, lo que supone una ventaja ya que el proyecto se desarrolla en C++. Tiene una gran cantidad de funciones relacionadas con las técnicas de visión artificial necesarias para el desarrollo del programa.

- Dynamsoft Barcode Reader

  Es la opción elegida para el reconocimiento de los códigos de barras. Tras estudiar gran cantidad de posibilidades para esta tarea se encuentra esta opción como la más sencilla, tanto de integrar como de utilizar, al ser enormemente intuitiva y con ejemplos que ayudan al entendimiento.

Capítulo 2 ESTADO DE LA CUESTIÓN

2.1 INTRODUCCIÓN HISTÓRICA

Los sistemas de visión artificial o visión por computador aparecieron en los años 60 con el fin de permitir a los ordenadores el entendimiento del mundo real tal y como lo entendemos los humanos a partir de imágenes. No será hasta las dos últimas décadas del siglo XX que estos sistemas no empiecen a perfeccionarse y a tener una utilidad real en la automatización de los procesos de la industria.

A lo largo de los años los sistemas de visión artificial han ido desarrollándose y evolucionando a la vez que aumentaban los campos de trabajo en los que era posible la implantación de dichos sistemas.

Los años e hitos más importantes a lo largo de la historia de la tecnología OCR son los siguientes.

En el año 1929 se concede en Alemania a Tauschek la primera patente por el método OCR, la cual también le sería concedida en el 1935 en EEUU. Dos años antes, en el año 1933 se concede en EEUU la primera patente por este método a Handel.

En el 1953 surge la primera máquina para trabajar con el método OCR, encargada al criptoanalista Harvey por la agencia nacional de seguridad de los EEUU tres años antes.

En la década de los años sesenta y setenta empieza a utilizarse en algunas empresas de forma frecuenta, sobre todo aquellas encargadas de la mensajería, y es en el año 1965 cuando da el salto a Europa empezando a utilizarse en Gran Bretaña.

En la actualidad los sistemas OCR se han perfeccionado hasta alcanzar una tasa ínfima de errores. Además están presente en todo tipo de industrias y están disponibles para todos. A partir de esta tecnología han surgido otras como ICR (reconocimiento inteligente de caracteres) u OMR (reconocimiento óptico de marcas).

2.2 ESTADO DEL ARTE Y MOTORES OCR

Aunque es posible pensar que la información dejará de existir de una manera física que no sea a través de una pantalla, en lo presente esa realidad aún parece lejana y por ello precisamos de mecanismos que sean capaces de reconocer esa información para realizar tareas derivadas. Debido al uso masivo y continuo de cualquier tipo de documentos en formato papel es necesario un sistema que reconozca tanto la forma o dimensión física del documento, como el significado de su contenido para un posterior análisis.

Existen muchas aplicaciones de estos sistemas de visión como son reconocimiento y análisis de objetos, mediciones espaciales, o reconocimiento y procesamiento de imágenes entre otras. Dentro de ésta última se encuentra el reconocimiento óptico de caracteres que es el método que utilizamos en este proyecto.

Hoy en día ya hay desarrollados multitud de motores para aplicaciones que usan el método OCR.

Alguno de los estos aplicaciones son los siguientes:

- Free OCR: se trata de uno de estos motores más fáciles y disponibles para su uso. Se trata de un software para Windows completamente gratuito que incluye incluso una versión online.

- Simple OCR: de la misma manera que el anterior es un software freeware, gratuito, con una licencia de uso comercial disponible para Windows.

- ExperVision TypeReader & OpenRTK: aplicación con software de licencia comercial disponible para los tres sistemas operativos más comunes. Ha recibido multitud de buenas valoraciones por revistas tecnológicas desde 1991 y es el único motor OCR que ganó varios años seguidos las pruebas de la Universidad de Nevada (UNLV).

- ABBY Finereader: es una de las aplicaciones más importantes para el reconocimiento texto mediante la técnica OCR. Tiene de una versión gratuita y de otra ampliada de ámbito profesional. Tiene una licencia comercial. Reconoce más de 190 lenguas e incluso códigos de barras. Está disponible para el sistema operativo de Windows. Muy utilizada para trabajar con interfaces específicas.

- GOCR: programa de reconocimiento de texto con OCR. Su uso es completamente gratuito y se encuentra disponible para los sistemas operativos de Windows, Linux y Mac Os. Fue de las primeras aplicaciones en desarrollarse.

- Tesseract: motor OCR originalmente desarrollado como software propietario pero que actualmente, desde 2005 es desarrollado por Google y se distribuye bajo licencia Apache. Funciona en 9 idiomas aunque puede ser entrenado para cualquier otro. Desde 1995 ya es considerado como uno de los mejores motores OCR y aunque al principio sus primeras versiones imponían varias restricciones como el tipo de imagen, se fue adaptando y gracias a la librería Leptónica y al desarrollo de Google, sigue estando en la cabeza de motores OCR.

- OmniPage: este software es de licencia comercial. Es utilizado por grandes empresas debido a la profesionalidad y el gran desarrollo del programa. Se encuentra disponible para Windows y Mac OS. Es un producto de Nuance Communications (empresa multinacional americana de tecnologías software).

2.3 ESTUDIO DE LAS HERRAMIENTAS EXISTENTES PARA EL DESARROLLO DEL PROYECTO

A continuación se comentan y analizan las posibilidades para cada uno de los recursos o herramientas necesarias para la realización de este proyecto.

2.3.1 ENTORNO DE DESARROLLO INTEGRADO (IDE)

El IDE es una herramienta de trabajo, una aplicación que ayuda y facilita al usuario el desarrollo de su programa o software. Ayuda a la elaboración del código, corrigiendo errores, reconociendo la sintaxis o depurando entre otras tareas. Tiene una interfaz gráfica para el usuario y suelen ser ofrecer un marco de trabajo para diferentes lenguajes de programación. Aunque hay una enorme cantidad de IDEs (CodeLite, Eclipse, MS Visual Studio, Netbeans…), a continuación se mencionan aquellos que han sido probados:

- Dev-C++: es un IDE bueno para la programación en C y C++, aunque esté programado en Delphi. Se trata de un software un tanto obsoleto, con pocas actualizaciones, sin constructor de interfaz gráfica ni análisis en estático. Todo esto sumado a la dificultad de integración de las librerías necesarias supusieron el descarte de esta opción.

- CodeLite: se trata de un software libre, con muchas ventajas entre las cuales se encuentra la familiaridad con dicho IDE pues ya ha sido utilizado anteriormente. Problemas con las actualizaciones a la vez que con la integración hicieron que se optará por otra alternativa.
- MS Visual Studio: finalmente este es el IDE elegido. La gran cantidad de ventajas como la personalización de la interfaz, la acepción de varios lenguajes además de C/C++. La universalidad del software, la enorme cantidad de trabajo e información en foros y páginas web hizo que no fuese ningún problema la adaptación a este software.

2.3.2 LIBRERÍAS DE VISIÓN ARTIFICIAL

Este tipo de librerías adquieren un papel fundamental en el desarrollo del proyecto puesto que dichas librerías contienen las funciones y técnicas necesarias para el tratamiento de imágenes el cual es indispensable para el reconocimiento posterior.

Como se trata de una tecnología en auge, hay gran cantidad de librerías desempeñan las mismas tareas y cada vez son más. Seguidamente se muestran las más analizadas.

- HALCON: se trata de unas librerías de visión artificial de ámbito profesional. Se puede implementar en programas de distintos lenguajes. Aunque profesionalmente se trate de una opción excepcional, en este proyecto no serán usadas puesto que hay una opción que se adapta mejor a las necesidades de este.
- OpenCV: se trata de unas librerías de visión artificial de ámbito libre, programadas en C y C++. Tiene un enfoque hacia las aplicaciones en tiempo real. Su gran cantidad de funciones y algoritmos que abarcan varias áreas de la visión artificial, así como su fácil acceso,

integración y documentación sobre ella es lo que hace que sea la librería elegida para el proyecto.

- SimpleCV: se trata de un conjunto de librerías de visión artificial entre las que se encuentra OpenCV. Está desarrollada en Python. Su dificultad a la hora de la instalación y a que lo necesario para el proyecto se encuentra en la parte de OpenCV, conlleva que no tenga sentido elegir esta alternativa.

2.3.3 SOFTWARE DE LECTURA DE CÓDIGOS DE BARRAS

En el proyecto, la información del documento, además de presentarse en forma legible debido a los caracteres del alfabeto números y símbolos de escritura también se presenta encriptada en códigos de barras en este caso de tipo PDF417. Este tipo de códigos se trata de códigos 2D los cuales pueden almacenar mucha más información en menos espacio que los de una sola dimensión. Dentro de los códigos de dos dimensiones los de tipo PDF417 tienen la ventaja de almacenar grandes cantidades de información, hasta 1600 caracteres, teniendo además la ventaja de que se pueden enlazar varios códigos aumentando así la cantidad de información. Esto los haces especialmente útiles para guardar grandes cantidades de texto, imágenes y cualquier otro tipo de datos. Entre las opciones analizadas para la lectura de estos códigos se destacan los mencionados a continuación.

- Zbar: se trata de una librería de lectura de códigos de barras muy extendida, programada tanto en C++ como en Python, entre otros lenguajes de programación. Aunque su descarga e implementación no resultan excesivamente complicados, faltan ficheros ".h" entre que se llaman a la hora de ejecución de los programas y los cuales hay que añadir manualmente, junto con nuevas librerías externas lo que dificulta la tarea de integración completa.

- Zxing: Es una librería para tratamiento de códigos de barras desarrollado en Java. Se trata de unas librerías extensas pero principalmente destinadas para la programación en Android lo que dificulta bastante su implementación en el proyecto y por ello no es la opción elegida.
- Dynamsoft Barcode Reader: aunque se trata de una aplicación ya elaborada para el tratamiento de códigos de barras, es la opción elegida puesto que contiene unas librerías propias perfectamente agrupadas, desarrolladas en C. Además la aplicación contiene varios ejemplos de programas que usan dichas librerías y que sirven de ejemplo y base para el desarrollo de la parte relacionada con los códigos de barras de este proyecto.

Capítulo 3 DESARROLLO Y RESULTADOS

3.1 ESTUDIO, ANÁLISIS Y MONTAJE DE LAS APLICACIONES Y ELEMENTOS NECESARIOS PARA EL DESARROLLO DEL PROYECTO

En esta primera parte del trabajo se indica cual son todos los elementos necesarios, que se utilizarán, para la realización y optimización del proyecto final, así como la manera de instalarlos, incluirlos e implementarlos para el correcto funcionamiento de nuestras aplicaciones.

En este proyecto se trabaja con Visual Studio como IDE (entorno de desarrollador integrado). A pesar de que muchas de los programas serán probados directamente en la ventana de comandos, la mayor parte del proyecto será programado en Visual Studio 2013.

Se elige Visual Studio debido a la facilidad para desarrollar en esta plataforma gracias a su fácil disposición y que su interfaz de trabajo está mejor estructurada y organizada que en otras aplicaciones similares como Codelite o DevC++ en las que se han realizado las mismas comprobaciones de funcionamiento de los primeros programas.

La versión de 2013 ha sido la elegida, ya que debido a que es una versión con unos años de antigüedad, nos aseguramos de que es una versión trabajada y perfeccionada, además de haber sido utilizada para proyectos similares. Por otro lado tras probar otras versiones como 2017 o 2015, encontramos en éstas mayor dificultad para incluir nuestras librerías de visión artificial.

En lo que respecta a las librerías de visión artificial se opta finalmente por OpenCV de entre las opciones comentadas anteriormente, y con ella se hará toda

la parte de tratamiento y procesamiento de las imágenes que sean utilizadas en el proyecto.

Esta decisión es tomada debido a que se trata de la librería que mejor se adapta a las necesidades de este proyecto. Esto se debe a que es una librería muy universal, de fácil acceso, con mucha información sobre su funcionamiento y que para el uso que se le va a dar en este proyecto hay mucho trabajo de la comunidad de internet que facilitan su entendimiento.

Por último en lo que se refiere al reconocimiento de los códigos de barras, se optará por utilizar Dynamsoft Barcode Reader para tratar los códigos PDF-417 en los que se encuentra la información que ayudará a un mejor reconocimiento de los documentos.

Los motivos son abundantes entre los que destacan su fácil implementación en código debido a un ejemplo que viene con la aplicación de descarga así como las dificultades que se han encontrado con la implementación e integración de otras librerías como ZXing o ZBar.

3.1.1 DESCARGA DEL IDE Y DE LAS LIBRERÍAS DE VISIÓN ARTIFICIAL Y DE CÓDIGOS DE BARRAS

1. Primero descargar Visual Studio 2013 que será el entorno de trabajo y en el cual se desarrollará todo el proyecto. Se puede obtener de manera directa desde la página web de visualstudio, en la pestaña de descargas, versiones anteriores, se descargará e instalará Visual Studio 2013 Comunnity Edition, la cual es gratuita y cuyas opciones por defecto de tanto de descarga como de instalación son adecuadas.
2. Descargar las librerías de visión artificial. En este caso se elegirá la versión 3.0.0 debido a que se han encontrado más dificultades de compatibilidad con versiones más recientes como la 3.3.0 o 3.2. Además la 3.0.0 es una versión perfeccionada y suficiente para la elaboración de

los programas necesarios. Para ello también se pueden obtener dichas librerías desde la página web de opencv de manera gratuita.

3. Crear una carpeta con el nombre "C:\OpenCV-3.3.0". Tras esto extraer el ejecutable descargado de la página web de opencv, cliquear dos veces sobre y seleccionar la carpeta recién creada como localización de destino de “extraer en”.

4. Descargar Dynamsoft Barcode Reader 6.0 para el reconocimiento de los códigos de barras. Para ello se podrá descargar de la página oficial de Dynamsoft, tanto si se quiere la versión completa como si se prefiere la prueba gratuita de 30 días. En el caso de este proyecto se opta por la opción gratuita de 30 días y la descarga se realiza desde la página web http://descargar.cnet.com desde la cual se puede realizar la descarga de manera más sencilla y sin la necesidad de introducir el correo de empresa que se requiere si se hace este paso desde la página oficial de Dynamsoft. Una vez descargado al ejecutar el programa aparecerá una pantalla como la siguiente:

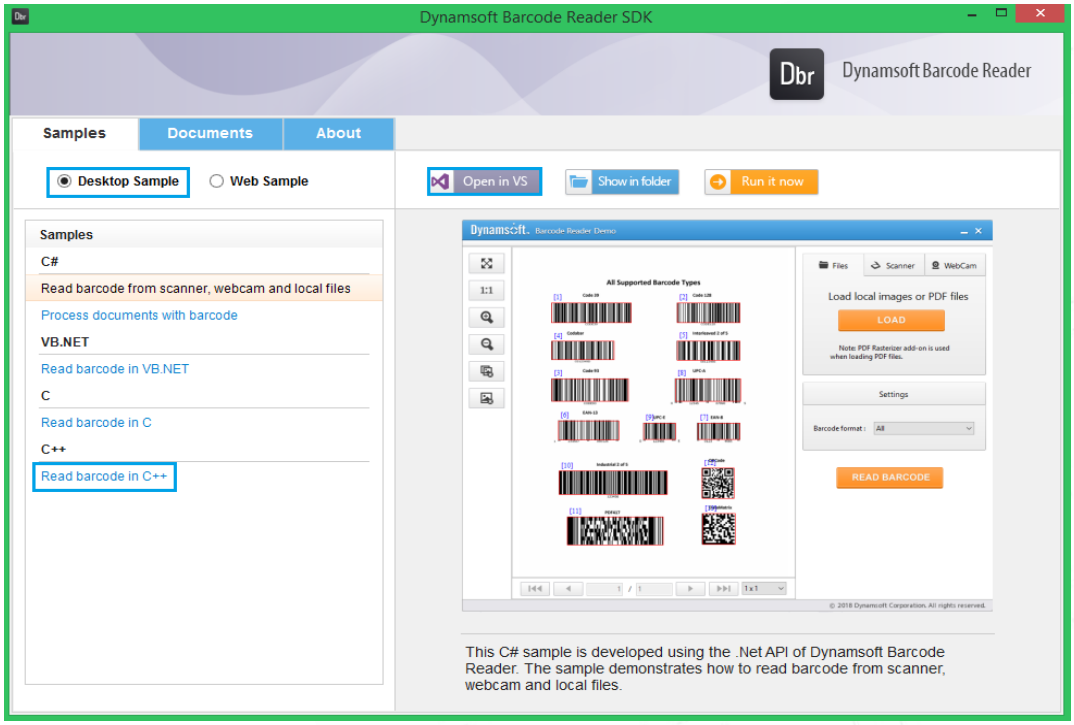


Figura 1. Pantalla de inicio de Dynamsoft Barcode Reader

En esta pantalla seleccionando "Desktop Sample", en la parte de "Read barcode in C++" y abriéndolo en Visual Studio se podrá ver un código de ejemplo que más tarde servirá como ejemplo y modelo para el reconocimiento de los códigos de barras.

3.1.2 PREPARACIÓN DEL IDE E INCORPORACIÓN DE LAS LIBRERÍAS DE VISIÓN ARTIFICIAL Y DE CÓDIGOS DE BARRAS

1. Añadir el directorio **bin** de nuestra versión de Visual Studio y OpenCV al **PATH** del sistema operativo. Si se siguen las indicaciones anteriores de este proyecto, se trabaja con Visual Studio 2013 y OpenCV-3.0.0 por lo que el directorio que se ha de incluir es "C:\OpenCV-3.0.0\opencv\build\x64\vc12\bin".

   Se elige **x64** puesto que estamos trabajando con la versión de 64 bits y **vc12** hace referencia a la versión de Visual Studio; si fuese la versión de 2015, por ejemplo, habría que añadir vc14.

   La manera de incluir este directorio es la siguiente:

   Acceder a panel de control, "Sistema", "Configuración avanzada del sistema", "Variables de entorno", y en variables del sistema seleccionar "PATH", "Editar", copiar el directorio a incluir, y finalizar dando a "Aceptar" y "Aplicar".

   A continuación se deja una imagen del camino o ruta para realizar este paso.

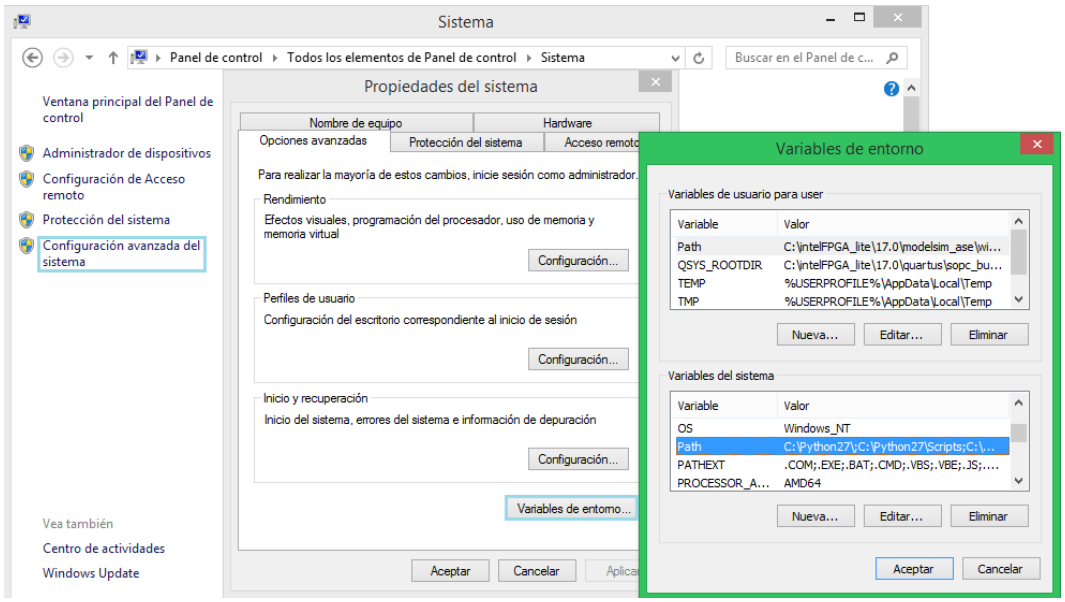


Figura 2. Camino hasta variables de entorno donde incluir el directorio

2. Abrir Visual Studio 2013, seleccionar File, New; Project.
3. En la parte de Visual C++ seleccionar "Empty Project" y elegir el nombre y la dirección deseada para nuestro programa.
4. En la ventana de "Solution Explorer" cliquear con el botón derecho sobre el nombre del proyecto, y luego seleccionar "Add" y "New Item" (también se puede seleccionar en la sección Project en la barra superior de menús). Tras esto seleccionar "C++ File (cpp)" y elegir el nombre deseado (preferiblemente que coincida con el nombre del proyecto).
5. Tras esto ya el siguiente paso es configurar las propiedades de nuestro proyecto, añadiendo los includes y las librerías necesarias para que el programa sea capaz de entender todas las funciones que posteriormente se utilicen.

   Para eso hay que seleccionar, en la barra superior de menús, "Project", "Project Properties".

En la parte de "Configuration Properties", en "VC++ Directories", seleccionar primero "Include Directories" y ahí pinchar en "Editar" e incluir la ruta de los includes que en el caso de este proyecto es:

- "C:\OpenCV-3.0.0\opencv\build\include", necesaria para todo lo respectivo a la imagen.
- "C:\Program Files (x86)\Dynamsoft\Barcode Reader 6.0\Components\C_C++\Include", necesaria para el tratamiento de los códigos de barras.

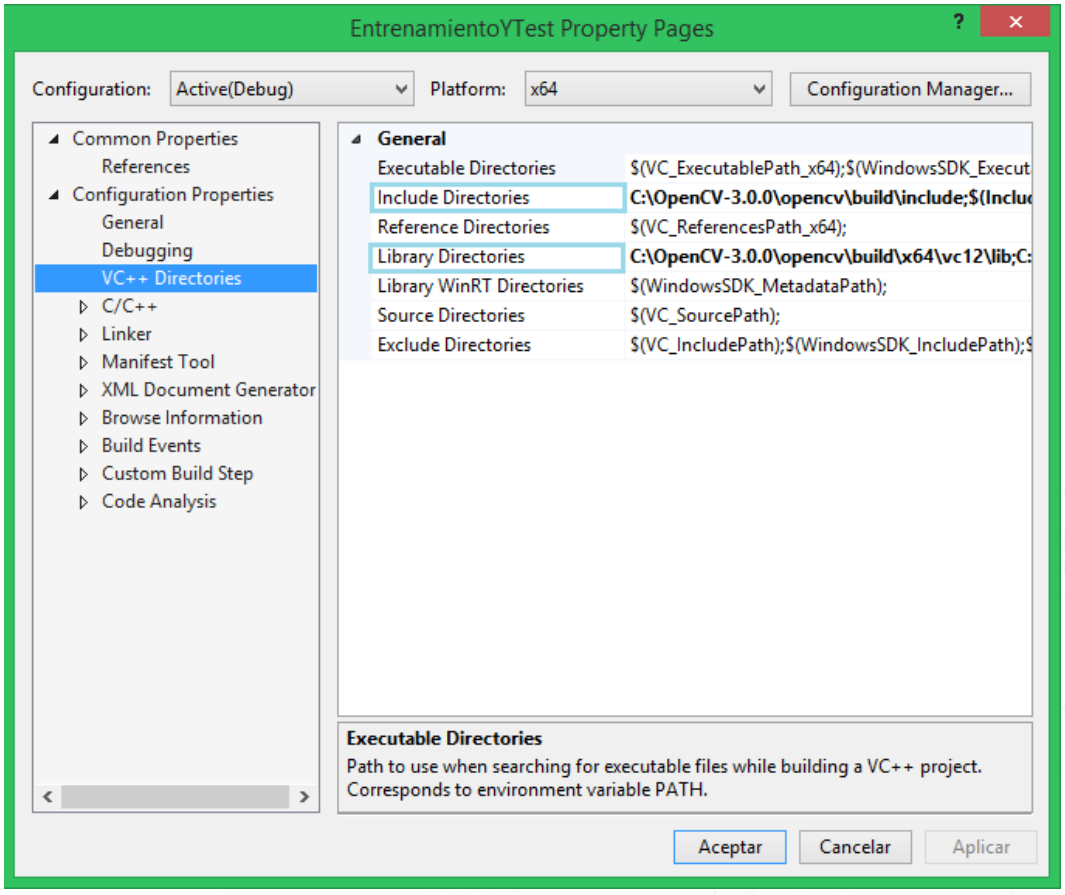


Figura 3. Incorporación de directorios para los includes y librerías

Posteriormente, en la parte también de "VC++ Directories" incluir, de manera análoga a los includes, en la parte de "Library Directories" la ruta de las librerías necesarias que en este caso es:

-"C:\OpenCV-3.0.0\opencv\build\x64\vc12\lib".

-"C:\Program Files (x86)\Dynamsoft\Barcode Reader 6.0\Components\C_C++\Redist\x64".

-"C:\Program Files (x86)\Dynamsoft\Barcode Reader 6.0\Components\C_C++\Lib".

Ahora hay que incluir las librerías necesarias. Como se va a trabajar en modo debug, hay que añadir aquellas que acaban en una "d". Estas se encuentran en la ruta de dirección que se acaba de añadir en "Library Diretories". Para añadirlas, hay que ir En la parte de "Configuration Properties" (dentro de "Project Properties") y ahí seleccionar en la parte de "Linker" en "Input", cliquear en "Additional Dependencies" y ahí copiar las librerías necesarias que son:

opencv_ts300d.lib

opencv_world300d.lib (en realidad solo hace falta ésta).

El numero 300 hace referencia a la versión de OpenCV (en este caso 3.0.0).

Si se trabajase en modo reléase serían las mismas librerías pero sin la "d" final.

Para el caso de las librerías necesarias para la parte de reconocimiento de códigos la que se debe incluir es la siguiente:

DBRx64.lib

Estas librerías que se acaban de mencionar (.lib) son llamadas librerías estáticas, las cuales se cargan al compilar el programa y se quedan en el ejecutable del programa, quedando en este una copia de dicha librería, lo que permite que se puedan usar incluso en un ordenador que no tenga dichas librerías si previamente fue ejecutado en uno que si las tuviera.

Por otro lado están las librerías dinámicas (.dll), que se cargan a la hora de ejecución del programa, y son necesarias en todo momento en el ordenador o instrumento sobre el que se trabaje. De este tipo es necesario que se incluyan tres, necesarias para la parte de los códigos de barra y son las siguientes:

DynamicPdfx64.dll

DynamsoftBarcodeReaderx64.dll

vcomp110.dll

La manera de incluir este tipo de librerías es añadiéndolas en la carpeta donde se está desarrollando el proyecto.

6. Finalmente antes de comenzar en la parte de "Build" en la barra de menús, en "Configuration manager", comprobar que está en modo debug y en Active solution platform seleccionar "x64" (si no está por defecto hay que seleccionar "New" y en "Type or select the new platform debería de aparecer").

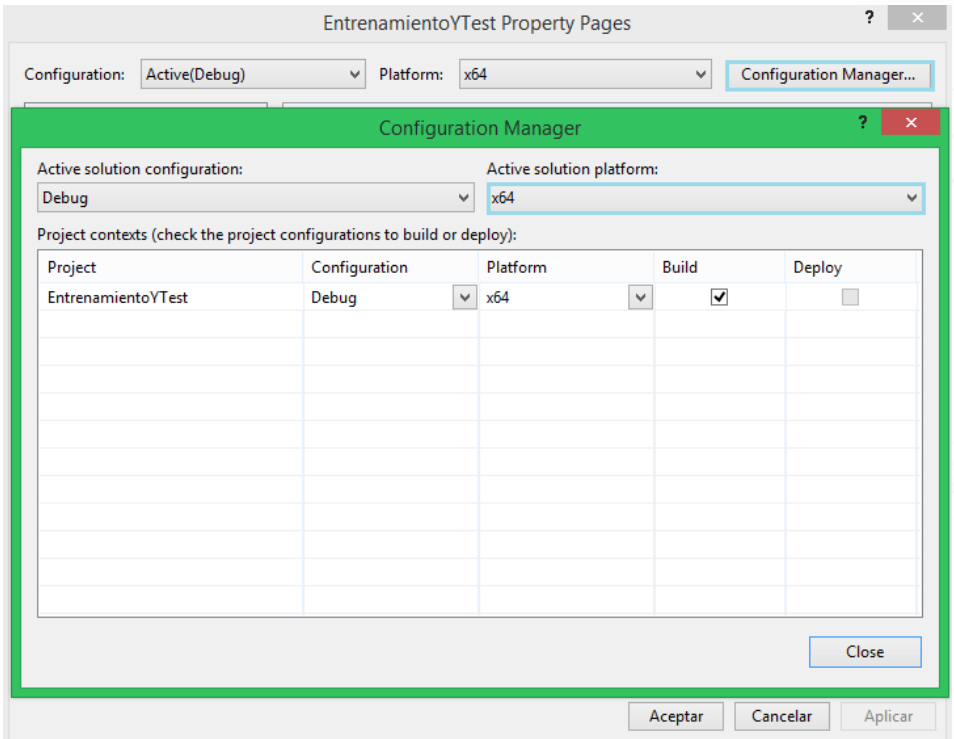


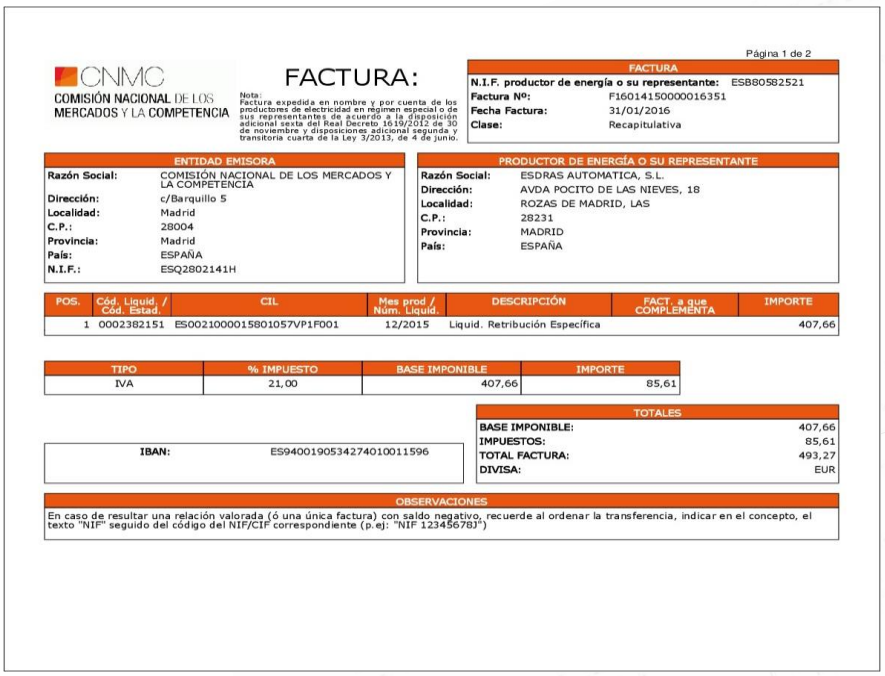
Figura 4. Configuration manager

3.2 PRUEBAS AISLADAS DE CARGADO DE IMÁGENES Y PRE-PROCESAMIENTO

En este apartado del trabajo, se explica, como el propio nombre indica, la manera en que se cargan las imágenes y los primeros tratamientos que son necesarios aplicar para el posterior uso correcto de dicha imagen.

3.2.1 MÉTODO DE CARGADO DE IMÁGENES

En primer lugar cabe destacar que la imagen con la que se va a trabajar, ha de estar en la misma carpeta en la que está el proyecto en la que se va a utilizar y debe ser llamada desde el programa de la misma manera en que esté guardada dicha imagen. Aunque en los distintos apartados de este proyecto se utilizan distintas imágenes para comprobar funcionamientos intermedios, la imagen sobre la que se realizará el reconocimiento final del documento es la siguiente:



Página 1 de 2

CNMC
COMISIÓN NACIONAL DE LOS MERCADOS Y LA COMPETENCIA

FACTURA:

Nota: Factura expedida en nombre y por cuenta de los productores de electricidad en régimen especial o de sus representantes de acuerdo a la disposición adicional sexta del Real Decreto 1619/2012 de 30 de noviembre y disposiciones adicional segunda y transitoria cuarta de la Ley 3/2013, de 4 de junio.

FACTURA	
N.I.F. productor de energía o su representante:	ESB80582521
Factura Nº:	F16014150000016351
Fecha Factura:	31/01/2016
Clase:	Recapitulativa

ENTIDAD EMISORA	
Razón Social:	COMISIÓN NACIONAL DE LOS MERCADOS Y LA COMPETENCIA
Dirección:	c/Barquillo 5
Localidad:	Madrid
C.P.:	28004
Provincia:	Madrid
País:	ESPAÑA
N.I.F.:	ESQ2802141H

PRODUCTOR DE ENERGÍA O SU REPRESENTANTE	
Razón Social:	ESDRAS AUTOMATICA, S.L.
Dirección:	AVDA POCITO DE LAS NIEVES, 18
Localidad:	ROZAS DE MADRID, LAS
C.P.:	28231
Provincia:	MADRID
País:	ESPAÑA

POS.	Cód. Liquid. / Cód. Estad.	CIL	Mes prod / Núm. Liquid.	DESCRIPCIÓN	FACT. a que COMPLEMENTA	IMPORTE
1	0002382151	ES0021000015801057VP1F001	12/2015	Liquid. Retribución Específica		407,66

TIPO	% IMPUESTO	BASE IMPONIBLE	IMPORTE
IVA	21,00	407,66	85,61

TOTALES	
BASE IMPONIBLE:	407,66
IMPUESTOS:	85,61
TOTAL FACTURA:	493,27
DIVISA:	EUR

IBAN: ES9400190534274010011596

OBSERVACIONES

En caso de resultar una relación valorada (ó una única factura) con saldo negativo, recuerde al ordenar la transferencia, indicar en el concepto, el texto "NIF" seguido del código del NIF/CIF correspondiente (p.ej: "NIF 12345678")

Figura 5. Imagen principal del proyecto

Un primer programa de cargado de imágenes en el que se lee un archivo de imagen y se muestra por ventana de comandos es el siguiente.

```
#include <cstdio>
#include <C:\OpenCV-3.0.0\opencv\build\include\opencv2\opencv.hpp>
#include <C:\OpenCV-3.0.0\opencv\build\include\opencv2\core\core.hpp>

using namespace cv;

int main()
{
	Mat img = imread("imagen1.jpg"); // imagen a leer

	if (!img.data)  //comprobamos que la imagen se haya leído
correctamente
	{
		std::cout << "Imagen no cargada \n";
		//mostrando un mensaje de error en caso
		//de que haya habido cualquier error
		System("pause");
		return -1;

	}
	imshow("imagen prueba", img);		//mostramos la imagen
	waitKey();			//necesario para poder ver la imagen
	return 0;
}
```

Este programa aunque es muy sencillo es útil para saber si la incorporación de las librerías de visión artificial se ha hecho de manera adecuada y funcionan correctamente.

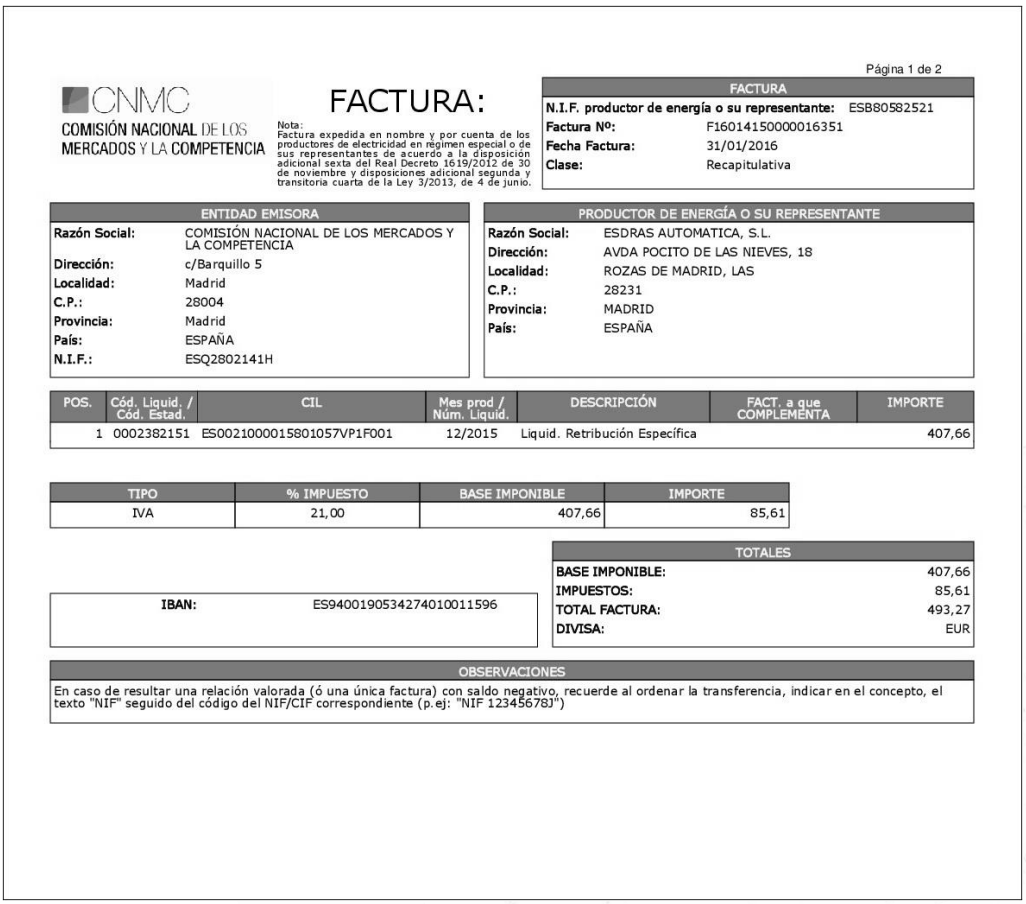
Como se puede apreciar, se utiliza el lenguaje C++ y lo primero que se encuentra en el programa son los "includes" necesarios para poder usar las funciones de lectura y muestreo de imágenes.

3.2.2 PRE-PROCESAMIENTO DE IMÁGENES

Tras este primer programa ahora se procede a implementar las primeras funciones de pre-procesamiento de imágenes. A pesar de que se pueden aplicar infinidad de

método de tratamiento de imágenes y repetirlos para obtener una mejor imagen de mayor calidad, en esta sección se explican dos de las más básicas y necesarias.

La primera es la conversión a escala de grises de una imagen. La imagen tratada se convierte, en función de la intensidad luminosa, a una escala de 8 bits en la que el valor 0 se considera color negro y el 255 color blanco. Puesto que se trata de una imagen digital, dividida en pixeles, en función de la luminosidad correspondiente a cada píxel se la asigna un valor entre 0 y 255 que significa un valor de gris proporcional, siendo el 255 para el píxel completamente blanco y 0 para el negro, quedando la imagen como se muestra a continuación.



Página 1 de 2

CNMC
COMISIÓN NACIONAL DE LOS MERCADOS Y LA COMPETENCIA

FACTURA:

Nota:
Factura expedida en nombre y por cuenta de los productores de electricidad en régimen especial o de sus representantes de acuerdo a la disposición adicional sexta del Real Decreto 1619/2012 de 30 de noviembre y disposiciones adicional segunda y transitoria cuarta de la Ley 3/2013, de 4 de junio.

FACTURA	
N.I.F. productor de energía o su representante:	ESB80582521
Factura Nº:	F16014150000016351
Fecha Factura:	31/01/2016
Clase:	Recapitulativa

ENTIDAD EMISORA	
Razón Social:	COMISIÓN NACIONAL DE LOS MERCADOS Y LA COMPETENCIA
Dirección:	c/Barquillo 5
Localidad:	Madrid
C.P.:	28004
Provincia:	Madrid
País:	ESPAÑA
N.I.F.:	ESQ2802141H

PRODUCTOR DE ENERGÍA O SU REPRESENTANTE	
Razón Social:	ESDRAS AUTOMATICA, S.L.
Dirección:	AVDA POCITO DE LAS NIEVES, 18
Localidad:	ROZAS DE MADRID, LAS
C.P.:	28231
Provincia:	MADRID
País:	ESPAÑA

POS.	Cód. Liquid. / Cód. Estad.	CIL	Mes prod / Núm. Liquid.	DESCRIPCIÓN	FACT. a que COMPLEMENTA	IMPORTE
1	0002382151	ES0021000015801057VP1F001	12/2015	Liquid. Retribución Específica		407,66

TIPO	% IMPUESTO	BASE IMPONIBLE	IMPORTE
IVA	21,00	407,66	85,61

TOTALES	
BASE IMPONIBLE:	407,66
IMPUESTOS:	85,61
TOTAL FACTURA:	493,27
DIVISA:	EUR

IBAN: ES9400190534274010011596

OBSERVACIONES

En caso de resultar una relación valorada (ó una única factura) con saldo negativo, recuerde al ordenar la transferencia, indicar en el concepto, el texto "NIF" seguido del código del NIF/CIF correspondiente (p.ej: "NIF 12345678J")

Figura 6. Imagen del proyecto en escala de grises

La función que hace dicha operación es la siguiente.

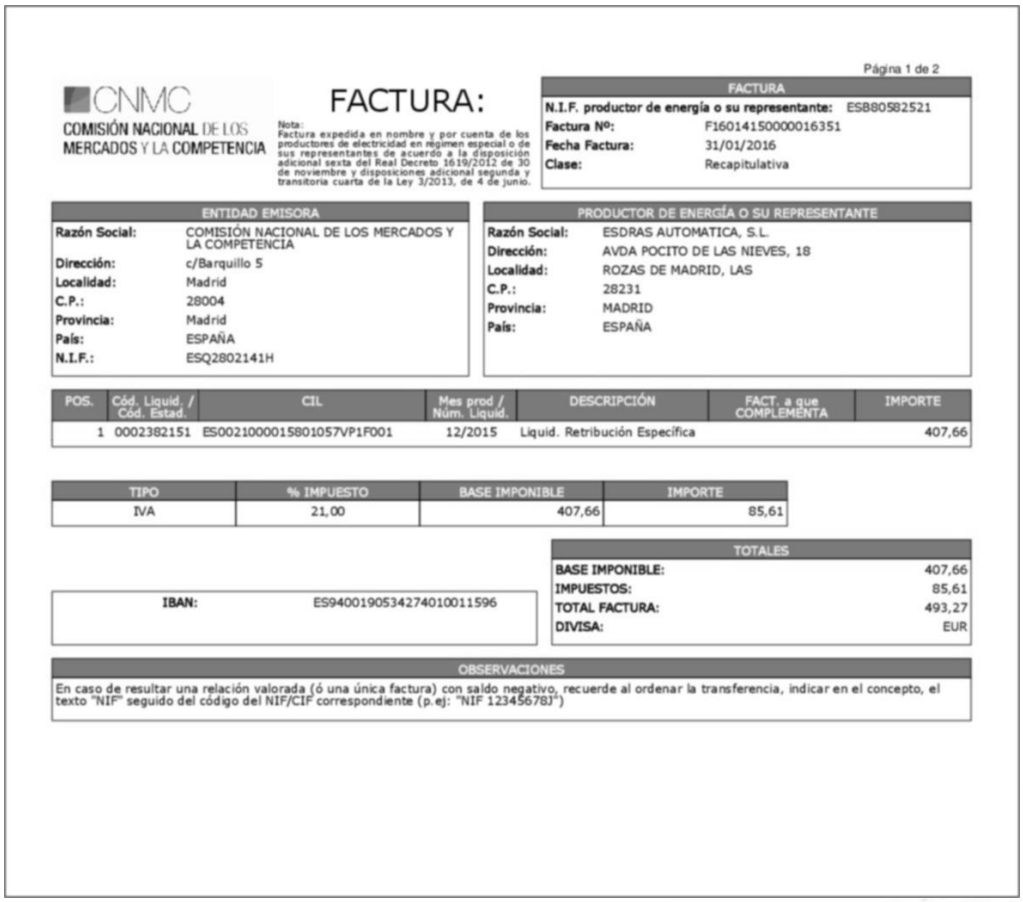
```
cv::cvtColor(img, imgGrayscale, CV_BGR2GRAY);            // convertir a
escala de grises
```

El "cv::" inicial es necesario para indicar al programa que se trata de una función de la librería de OpenCV (en el programa anterior no se utiliza delante de las funciones porque se declara, antes de iniciar el programa principal o main, que se va a utilizar el espacio de nombres cv). Los elementos de la función son "img" que se la matriz donde está guardada la imagen original, "imgGrayscale" que es la matriz donde se va a guardar la matriz en escala de grises y "CV_BGR2GRAY" que indica que se cambie de escala BGR (azul, verde y rojo) a escala de grises.

El siguiente proceso es la elección del valor umbral, esto es, elegir una frontera en esa escala de grises a partir de la cual los valores que estén por encima de dicho valor serán considerados como blancos y los que estén por debajo se les asignará el valor 0, es decir negro. Lo ideal para la imagen sería que todo lo que interesa, esté a un lado de esa frontera y lo que no interesa al otro lado. En las imágenes es difícil elegir ese umbral puesto que dichas imágenes no son perfectas, tienen sombras faltas de luminosidad o contraste entre otros inconvenientes. En esta parte inicial del proyecto se trabaja con imágenes razonablemente buenas, y lo único que se hace antes de elegir el valor umbral es aplicar un emborronamiento gaussiano que hace que el valor de gris de cada pixel siga una media gaussiana entre su valor y los de los pixeles adyacentes. Optamos por el que sigue la media gaussiana puesto que da un resultado más eficaz para las posteriores funciones, que el que se consigue con un filtro de mediana.

```
cv::GaussianBlur(imgGrayscale,              // img entrada
                 imgBlurred,                // img salida
                 cv::Size(5, 5),
// suavizamos el alto y ancho de la imagen según el valor de sigma
             0);                                        // valor de sigma,
que dice cuanto ha de ser emborronarse, cero hace que la función elija
el valor de sigma
```

La imagen de manera difuminada tras este emborronamiento es:



Página 1 de 2

CNMC
COMISIÓN NACIONAL DE LOS MERCADOS Y LA COMPETENCIA

FACTURA:

Nota:
Factura expedida en nombre y por cuenta de los productores de electricidad en régimen especial o de sus representantes de acuerdo a la disposición adicional sexta del Real Decreto 1619/2012 de 30 de noviembre y disposiciones adicional segunda y transitoria cuarta de la Ley 3/2013, de 4 de junio.

FACTURA	
N.I.F. productor de energía o su representante:	ESB80582521
Factura Nº:	F16014150000016351
Fecha Factura:	31/01/2016
Clase:	Recapitulativa

ENTIDAD EMISORA	
Razón Social:	COMISIÓN NACIONAL DE LOS MERCADOS Y LA COMPETENCIA
Dirección:	c/Barquillo 5
Localidad:	Madrid
C.P.:	28004
Provincia:	Madrid
País:	ESPAÑA
N.I.F.:	ESQ2802141H

PRODUCTOR DE ENERGÍA O SU REPRESENTANTE	
Razón Social:	ESDRAS AUTOMATICA, S.L.
Dirección:	AVDA POCITO DE LAS NIEVES, 18
Localidad:	ROZAS DE MADRID, LAS
C.P.:	28231
Provincia:	MADRID
País:	ESPAÑA

POS.	Cód. Liquid. / Cód. Estad.	CIL	Mes prod / Núm. Liquid.	DESCRIPCIÓN	FACT. a que COMPLEMENTA	IMPORTE
1	0002382151	ES0021000015801057VP1F001	12/2015	Liquid. Retribución Específica		407,66

TIPO	% IMPUESTO	BASE IMPONIBLE	IMPORTE
IVA	21,00	407,66	85,61

TOTALES	
BASE IMPONIBLE:	407,66
IMPUESTOS:	85,61
TOTAL FACTURA:	493,27
DIVISA:	EUR

IBAN: ES9400190534274010011596

OBSERVACIONES

En caso de resultar una relación valorada (ó una única factura) con saldo negativo, recuerde al ordenar la transferencia, indicar en el concepto, el texto "NIF" seguido del código del NIF/CIF correspondiente (p.ej: "NIF 12345678J")

Figura 7. Imagen principal difuminada tras emborronamiento gaussiano

Por lo tanto nuestra función para hacer que la imagen ya quede preparada para su procesamiento posterior queda de la siguiente manera.

```
cv::adaptiveThreshold(imgBlurred,                    // img entrada
                      imgThresh,                     // img salida
                      255,                           // valor de los
      pixeles completamente blancos
              cv::ADAPTIVE_THRESH_GAUSSIAN_C,        // usamos filtro
gaussiano en vez de la media
              cv::THRESH_BINARY_INV,                 // invertimos negros
y blancos
              11,                                    // tamaño de los
pixeles vecinos para hallar el valor umbral
              2);                                     // constante que se
resta a la media o media ponderada
```

Además de lo ya mencionado cabe destacar como se hace una inversión de la imagen binaria, es decir, una vez que ya se le ha asignado el valor negro o blanco a cada uno de los píxeles de la imagen, se invierten dichos valores. Esto se hace debido a que en las imágenes que originales generalmente tiene menos intensidad luminosa (cercanía al negro) los elementos que se quieren resaltar y sin embargo para el tratamiento digital de imágenes es más sencillo trabajar con blancos como elementos de interés.

Así pues la imagen sobre la que se realiza el procesamiento es la siguiente:

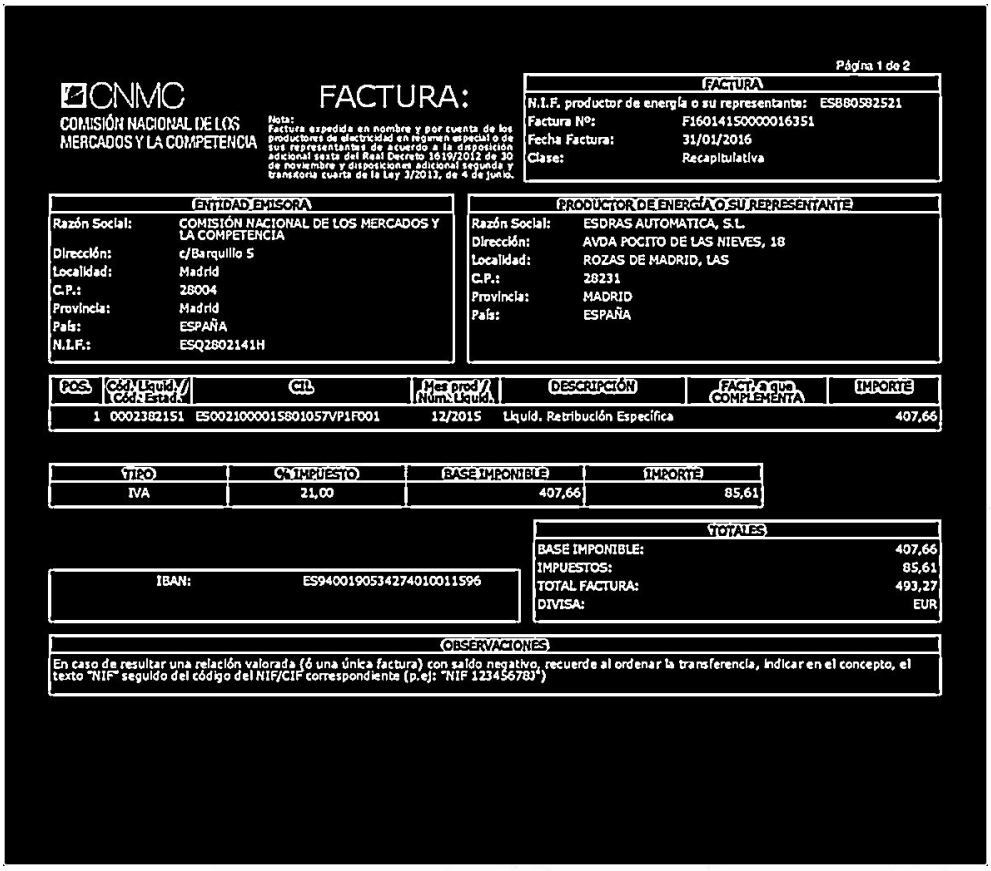


Figura 8. Imagen principal invertida tras el método del valor umbral

3.3 PRUEBAS AISLADAS DE PROCESAMIENTO DE IMÁGENES

En este apartado del proyecto lo que se va a realizar son ya técnicas de procesamiento de imágenes, con el fin de poder utilizarlas con y para el fin que se desee.

Las funciones de pre-procesamiento de imágenes explicadas en el apartado anterior también se incluyen en este apartado puesto que al fin y al cabo es también tratamiento de la imagen.

3.3.1 PREPARACIÓN DE LA IMAGEN PARA EL RECONOCIMIENTO DE CARACTERES

Todos estos procesos de tratamiento de imagen se requieren en nuestro caso para que el programa sea capaz de identificar los caracteres de la imagen de la manera adecuada. Para ello primero hay que saber qué es lo que está recibiendo el programa como imagen del carácter a reconocer y luego hay que saber si el reconocimiento es el adecuado (del carácter original).

El proceso de reconocimiento de caracteres que se va a usar en este proyecto es mediante el método KNN (k-vecinos más próximos), el cual se explica más adelante.

Primero para enviar la imagen a reconocer al programa, esta ha de sufrir una serie de transformaciones para que se identifiquen correctamente en ella las características significativas que harán que se reconozca de manera correcta.

Además de todas las funciones explicadas anteriormente (conversión a escala de grises, emborronamiento gaussiano y elección del valor umbral) hacen falta otra serie de técnicas, especialmente destinadas a seleccionar el rectángulo delimitador de cada carácter de manera adecuada sin que se desfigure excesivamente la imagen original.

Para conseguir este objetivo, lo primero es hacer una operación morfológica. Las operaciones morfológicas de las imágenes binarias consisten en una transformación de la imagen original comparándola con un elemento estructurador, con el fin de que en la imagen resultante se preserven y resalten las características más significativas de una imagen y eliminen aquellas que no interesan. Hay muchas operaciones morfológicas (unión, dilatación, erosión…) y es importante elegir bien tanto la operación como el elemento estructurador y su tamaño con el fin de que con el fin de mejorar una parte de la imagen, no se pierdan otros componentes significativos en otras zonas de la imagen.

En este proyecto después de hacer varias pruebas, se ha optado por elegir la operación morfológica de cierre que consiste en una dilatación seguida por una erosión. Como elementos estructuradores, generalmente se puede elegir entre un forma rectangular o elíptica, y en función de que sea mejor para el programa se elegirá entre uno u otro elemento (en el ejemplo mostrado a continuación se ha optado por un rectángulo).

```
cv::Mat grad3; //imagen resultante
cv::Mat morphKernel3 = cv::getStructuringElement(cv::MORPH_RECT,
cv::Size(1, 17));          //elección del elemento estructurador
cv::morphologyEx(imgThresh, grad3, cv::MORPH_CLOSE, morphKernel3);

//imgThresh (original) grad3(resultante)
```

En función de la imagen tratada y de los requerimientos necesarios de esta, se modificaría el tipo de elemento estructurador como su tamaño. También es importante saber dónde implementar esta función puesto que puede tener distintas utilidades: así como después del método del valor umbral puede ayudar a unir regiones inconexas para que se traten como una sola, de igual manera antes de la transformación a escala de grises puede servir para mejorar el contraste de la imagen.

```
cv::Mat imgThreshCopy;
imgThreshCopy = grad3.clone();         // hacemos copia de grad3 porque
findcountours modifica la imagen

      cv::findContours(imgThreshCopy,             // img entrada, estar
seguros de que es una copia
            ptContours,                           // contornos de
salida
            v4iHierarchy,                         // jerarquía de
salida
            cv::RETR_EXTERNAL,                    // recupera los
contornos exteriores solo
            cv::CHAIN_APPROX_SIMPLE);             // compresión
horizontal, vertical, y diagonal de los segmentos y dejamos solo los
puntos finales
```

La siguiente operación que hay que hacer es encontrar los contornos delimitadores de cada uno de los caracteres. Cabe destacar que la función utiliza para esta tarea modifica la imagen original por lo que hay que hacer una copia antes de llamar a dicha función. El resto de cosas necesarias para entender el funcionamiento de la función se explica en los comentarios del propio programa.

Tras explicar estas últimas funciones, se pasará a explicar el proceso de reconocimiento de caracteres mediantes el método KNN.

3.3.2 MÉTODO KNN - ENTRENAMIENTO

En este apartado se explica en que consiste este método, la forma en la cual se implanta en forma de código dentro de nuestro programa y las ventajas y limitaciones que éste presenta.

El método KNN (k vecinos más próximos) es un método de clasificación "perezoso" (lazy) puesto que requiere de un entrenamiento previo y la

clasificación posterior se basa en la similitud o distancia con los ejemplos introducidos en dicho entrenamiento.

El entrenamiento consiste en que, para cada dato posible, se introducen una serie de ejemplos representantes de dicho elemento de interés, esto es, se introduce una serie de ejemplos en forma de imágenes y se le dice al ordenador que objeto/carácter representa dicha imagen introducida. Se generan, tras el entrenamiento, paralelamente, dos estructuras de clasificación; 1. Un conjunto de datos (en este caso números que representan el código ASCII) que identifican a cada una de las imágenes y 2. Un conjunto de imágenes.

Para realizar este entrenamiento, por tanto, lo primero que se necesita es una serie de ejemplos con los datos posibles y para ello se ha elegido una imagen que contenga los elementos que sean de interés para el proyecto. En este caso los datos con los que se trabajará son letras mayúsculas y minúsculas, números, y principales signos de puntuación, todos ellos recogidos en la siguiente imagen.

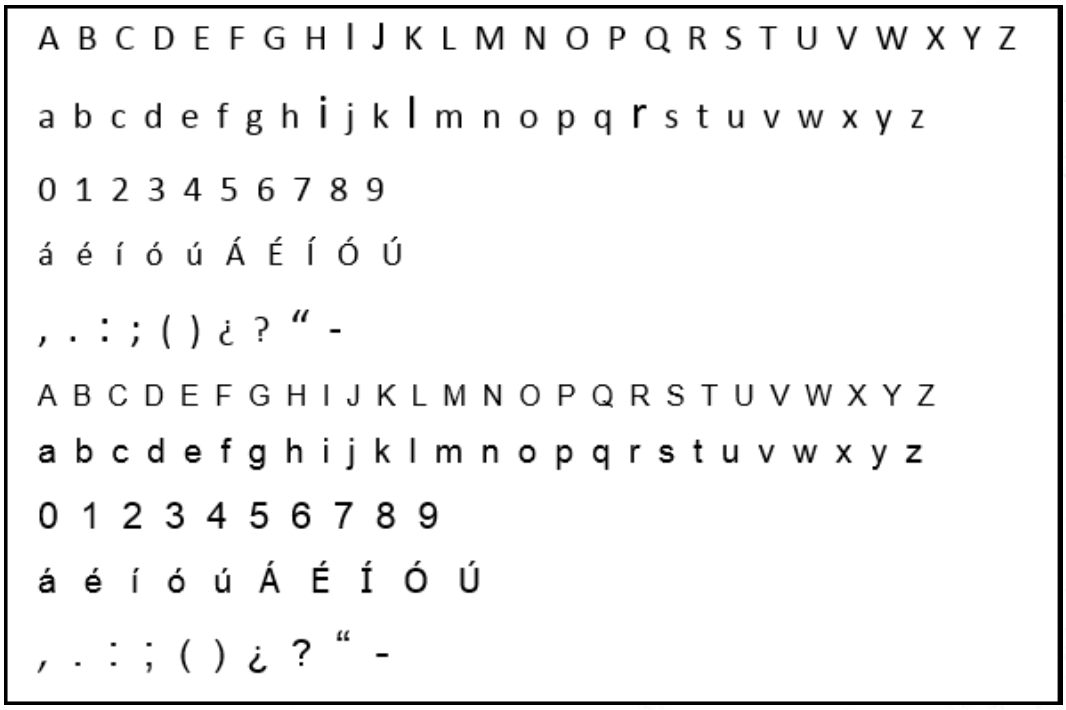


Figura 9. Imagen de entrenamiento de caracteres

Lo que hará el programa será preguntar por cada carácter de la imagen seleccionándolos por contorno y preguntar al usuario, entrenador, que introduzca por teclado de que carácter se trata (en esto consiste el entrenamiento). Cuantos más ejemplos se introduzcan mayor será la robustez del programa pues tendrá más imágenes con las que comparar; esto también hace que el programa se ralentice debido a que a mayor cantidad de datos, mayor costo computacional. En esta primera parte con introducir dos ejemplos por cada carácter de interés es suficiente para obtener un resultado razonablemente bueno.

En este programa ya se incluyen el procesamiento de imágenes comentado en las secciones anteriores, con el fin de preparar la imagen para que sea bien reconocida por el ordenador.

El programa de entrenamiento completo y comentado se muestra en la sección de código del final, pero algunas cosas que destacar son las siguientes:

```
	cv::Mat matClassificationInts;
	cv::Mat matTrainingImagesAsFlattenedFloats;

	// posibles caracteres en los que estamos interesados y que serán 
introducidos por teclado
	std::vector<int> intValidChars = { '0', '1', '2', '3', '4', '5', 
'6', '7', '8', '9',
		'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J',
		'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T',
		'U', 'V', 'W', 'X', 'Y', 'Z',
		'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j',
		'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't',
		'u', 'v', 'w', 'x', 'y', 'z',
		',', '.', ':', ';', '(', ')', '¿', '?', 'º', 'ª', '"', '-
', '@'};
```

En primer lugar ver como se crean dos matrices (“matClassificationInts” y “matTrainingImagesAsFlattenedFloats”), en las cuales se guardarán las estructuras de datos mencionadas anteriormente.

A continuación se ver cómo los caracteres aceptados por el programa al inicio del programa deben ser los mismos que los que hay en la imagen de entrenamiento para que cada vez que pregunte por uno de ellos se pueda responder de la manera

adecuada introduciendo el carácter correspondiente. Si lo introducido por teclado no está entre esos caracteres no será reconocido.

```
int intChar = cv::waitKey(0);
if (intChar == 27) {        // si pulamos esc
                            return(0);              // salir
                    }
                    else if (std::find(intValidChars.begin(),
intValidChars.end(), intChar) != intValidChars.end()) {     // si es
cualquier otra cosa de la lista que queriamos

                            matClassificationInts.push_back(intChar);
// añadir el carácter de clasificación a la lista entera de caracteres

                            cv::Mat matImageFloat;
//añadir la imagen de entranamiento (se necesitan primero algunas
coversiones) . . .
                            matROIResized.convertTo(matImageFloat,
CV_32FC1);       // cobertir de Mat a Float

                            cv::Mat matImageFlattenedFloat =
matImageFloat.reshape(1, 1);       // aplanar


      matTrainingImagesAsFlattenedFloats.push_back(matImageFlattenedFlo
at);       // añadir a Mat como si fuera un vector, esto es necesario
                            // porque es el tipo de datos que
KNearest.train acepta
                    }   // salir del if
```

En esta parte del programa lo que se realiza es leer lo introducido por teclado por el usuario para cada carácter que se le pregunta y si está entre los posibles, incorporarlo a la matriz "matClassificationInts" la correspondiente imagen de dicho carácter a la matriz "matTrainingImagesAsFlattenedFloats".

```
      cv::FileStorage fsClassifications("classifications.xml",
cv::FileStorage::WRITE);          // abrir la carpeta de
clasificaciones

      if (fsClassifications.isOpened() == false) {
// si no se abre correctamente
            std::cout << "error, incapaz de abrir el archive de
clasificaciones de entrenamiento";        // mostrar mensaje de error
            return(0);
// salir
      }
```

```
fsClassifications << "classifications" << matClassificationInts;
// escribir clasificaciones en la sección de clasificaciones del archivo
de clasificaciones
	fsClassifications.release();
// cerrar la carpeta de clasificaciones

	// guardar la imagenes de entranamiento en un archivo
//////////////////////////////////////////////////////

	cv::FileStorage fsTrainingImages("images.xml",
cv::FileStorage::WRITE);	// abrir el archivo de imagenes de
entrenamiento

	if (fsTrainingImages.isOpened() == false) {
//si no se abre
		std::cout << "error, incapaz de abrir el archive de
imagenes de entrenamiento, saliendo del programa\n\n";
// error
		return(0);
// salir
	}

	fsTrainingImages << "images" <<
matTrainingImagesAsFlattenedFloats;	// escribir imágenes de
entrenamiento en la sección de imágenes del archivo de imágenes
	fsTrainingImages.release();
// cerrar el archivo de imagenes de entranamiento
```

Finalmente se ve como se guardan los datos en dos archivos .xml: "classifications.xml" y "images.xml". Estas son las estructuras de datos de los que se habla anteriormente en los cuales se guardan el código ASCII y las imágenes identificadoras de los caracteres, las cuales se utilizarán en los programas de reconocimiento posteriores.

3.3.3 MÉTODO KNN – RECONOCIMIENTO DE CARACTERES

Tras haber hecho el entrenamiento propio del método KNN, el siguiente paso es el reconocimiento de caracteres usando este método y las estructuras de datos recientemente creadas. Para ello se probará con una imagen introducida por ordenador de la cual se intentara reconocer el texto escrito en la misma. Para realizar una primera verificación se ensayará con un texto sacado de la pantalla del ordenador, como el siguiente, para que esté perfectamente claro.

Figura 10. Imagen de prueba para el reconocimiento de caracteres

El programa completo de esta sección se encuentra también en la sección "Código" que se encuentra al final de este documento. En este programa lo que se hace es comprobar y comparar los caracteres de la imagen introducida con las estructuras de datos de imagen y carácter correspondientes, guardadas anteriormente (en el entrenamiento) en "images.xml" y "classifications.xml". Se comentan a continuación las partes más destacables:

```
class ContourWithData {
public:
	// variables miembro
///////////////////////////////////////////////////////////////////////////////
	std::vector<cv::Point> ptContour;		// contorno
	cv::Rect boundingRect;			// rectángulo
delimitador del contorno
	float fltArea;				// área del contorno
	///////////////////////////////////////////////////////////////////////////////
	bool checkIfContourIsValid() {				//
vemos si el contorno es valido
		if (fltArea < MIN_CONTOUR_AREA) return false;
		return true;
	}
	///////////////////////////////////////////////////////////////////////////////
	static bool sortByBoundingRectXPosition(const ContourWithData&
cwdLeft, const ContourWithData& cwdRight) {	// esta funcion nos
permite ordenar
		return(cwdLeft.boundingRect.x < cwdRight.boundingRect.x);
// los contornos de izquierda a derecha
	}
};
```

Lo primero que se hace en el programa de esta sección tras declarar la cabecera con los includes necesarios, es declarar la clase CountourWithData para comprobar si un determinado contorno encontrado es candidato de a contener un carácter de interés y la función para ordenar los caracteres de izquierda a derecha.

```
///////////////////////////////////////////////////////////////////////
int main() {
	std::vector<ContourWithData> allContoursWithData;           // 
declaramos vectores vacios
	std::vector<ContourWithData> validContoursWithData;         // 
que rellenaremos ordenadamente

	// leer en la entrenamiento de clasificaciones
/////////////////////////////////////////////////
	cv::Mat matClassificationInts;      // leeremos los números de 
clasificación en esta variable como si fuera un vector

	cv::FileStorage fsClassifications("classifications.xml", 
cv::FileStorage::READ);        // abrimos el archivo de clasificaciones

	if (fsClassifications.isOpened() == false) {
// si no se abre correctamente
		std::cout << "error, incapaz de abrir el archive de 
clasificaciones de entrenamiento, saliendo del programa\n\n";
// mostrar mensaje de error
		return(0);
// salir
	}

	fsClassifications["classifications"] >> matClassificationInts;
// leer la sección de clasificaciones en la variable de clasificaciones 
Mat
	fsClassifications.release();
// cerrar el archivo de clasificaciones

	// leer las imágenes de entrenamiento
/////////////////////////////////////////////////////////////

	cv::Mat matTrainingImagesAsFlattenedFloats;         // leeremos 
múltiples imágenes en esta única variable de imagen como si fuera un 
vector

	cv::FileStorage fsTrainingImages("images.xml", 
cv::FileStorage::READ);          // abrimos el archivo de entrenamiento 
de imagenes

	if (fsTrainingImages.isOpened() == false) {
// si no se abre correctamente
		std::cout << "error, incapaz de abrir el archive de las 
imagenes de entrenamiento file, saliendo del programa\n\n";
// mostrar mensaje de error
		return(0);
// salir
	}
	fsTrainingImages["images"] >> matTrainingImagesAsFlattenedFloats;
// leer la sección de imágenes en la variable imagen de entrenamiento 
Mat
	fsTrainingImages.release();
// cerrar el archivo de entrenamiento de imágenes
```

En esta segunda parte del programa lo que se hace es abrir y leer los archivos .xml en los que se encuentran las estructuras de datos con la información necesaria para el reconocimiento y guardarla en dos matrices para poder trabajar con ellas: es el paso inverso análogo al que se hacía en el apartado anterior de entrenamiento.

```
// entrenamiento
//////////////////////////////////////////////////////////////////////
	cv::Ptr<cv::ml::KNearest>  kNearest(cv::ml::KNearest::create());
// instanciar el objeto KNN

	// finalmente llegamos a la llamada para entrenar, tenga en
cuenta que ambos parámetros tienen que ser de tipo Mat(una única Mat)
	// aunque en realidad son múltiples imágenes / números
	kNearest->train(matTrainingImagesAsFlattenedFloats,
cv::ml::ROW_SAMPLE, matClassificationInts);
```

En esta parte de código que se encuentra justo encima lo que se está haciendo es instanciando el objeto KNN para poder realizar el reconocimiento con las estructuras de datos ya guardadas en dos matrices.

```
for (int i = 0; i < ptContours.size(); i++) {                // para cada
contorno
		ContourWithData contourWithData;
// instanciar un contorno con un objeto de datos
		contourWithData.ptContour = ptContours[i];
// asignar contorno al contorno con datos
		contourWithData.boundingRect =
cv::boundingRect(contourWithData.ptContour);          // obtener
rectangulo delimitador
		contourWithData.fltArea =
cv::contourArea(contourWithData.ptContour);                // calcular el
area delimitadora
		allContoursWithData.push_back(contourWithData);
// agregar contorno con el objeto de datos a la lista de todos los
contornos con datos
	}
for (int i = 0; i < allContoursWithData.size(); i++) {
// para todos los contornos
		if (allContoursWithData[i].checkIfContourIsValid()) {
// comprobar si es valido

	validContoursWithData.push_back(allContoursWithData[i]);
// si si, añadir a la lista de contornos validos
		}
	}
```

Esta parte de código sirve para una vez encontrados los contornos de la imagen ya procesada, comprobar si dichos contornos pueden contener algún carácter posible. En este caso la comprobación es simplemente una comparación por el tamaño del contorno.

```
	std::string strFinalString;		// declarar la cadena
final, esto será la secuencia final del programa

	for (int i = 0; i < validContoursWithData.size(); i++) {
// para cada contorno

		// dibujar un contorno verde alrededor del caracter
		cv::rectangle(matTestingNumbers,
// dibjar rectangulo en la imagen original
			validContoursWithData[i].boundingRect,		//
rectangulo a dibujar
			cv::Scalar(0, 255, 255),		// verde
			2);					// grosor

cv::Mat	matROI	=	matThresh(validContoursWithData[i].boundingRect);
// obtener la imagen ROI del rectángulo

v::Mat matROIResized;
		cv::resize(matROI, matROIResized,
cv::Size(RESIZED_IMAGE_WIDTH, RESIZED_IMAGE_HEIGHT));	//
redimensionar la imgen, esto será más consistente para el
reconocimiento y almacenamiento

		cv::Mat matROIFloat;
		matROIResized.convertTo(matROIFloat, CV_32FC1);
// convertir de Mat a float, necesario para llamar a find_nearest

		cv::Mat matROIFlattenedFloat = matROIFloat.reshape(1,
1);

		cv::Mat matCurrentChar(0, 0, CV_32F);

		kNearest->findNearest(matROIFlattenedFloat, 1,
matCurrentChar);	// ya podemos llamar a find_nearest !!!

		float fltCurrentChar =
(float)matCurrentChar.at<float>(0, 0);

		strFinalString = strFinalString +
char(int(fltCurrentChar));		// añadimos el caracter a la cadena
completa
	}
```

Finalmente lo que se hace es transformar o redimensionar nuestros contornos ya válidos a las dimensiones específicas apropiadas para poder llamar a la función findNearest para encontrar el vecino más próximo y así realizar el correcto reconocimiento. Por eso esto es tan importante un buen entrenamiento al igual que un buen pre-procesamiento de la imagen puesto que de eso dependerá de que se reconozca la imagen de una letra con un carácter u otro. Esto se hace para cada contorno y se van añadiendo los resultados a la cadena final.

El resultado final es el siguiente:

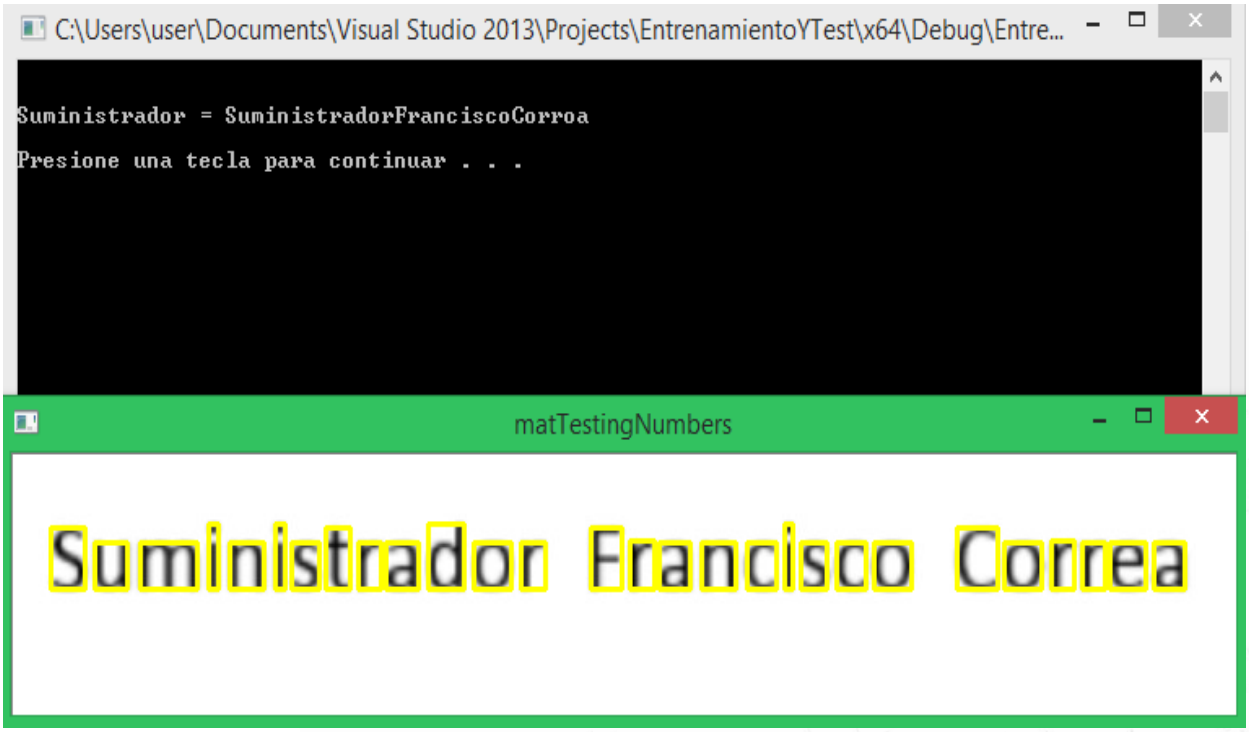


Figura 11. Resultado del reconocimiento de caracteres

Como se puede observar el reconocimiento no es perfecto puesto que la letra "e" es reconocida como una "o". Esto es debido a que la morfología utilizada para poder abarcar el "puntito" de la letra "i" era hacer una morfología con mucho peso

en la componente vertical lo que hace que la "e" se difuminase demasiado y se confundiera con la "o".

Este problema podrá ser corregido más adelante cuando se implante las primitivas de lenguaje de usuario y se tengan los datos sobre los posibles valores de las distintas secciones a reconocer.

3.4 PRUEBAS AISLADAS DE PROCESAMIENTO DE CÓDIGOS DE BARRAS

En este apartado del proyecto se explica el tratamiento de los códigos de barras y su posterior reconocimiento para sacar de ellos la información que ayudará a un mejor reconocimiento del documento.

3.4.1 CÓDIGOS PDF417 – PREPARACIÓN

El reconocimiento de los códigos no es una tarea complicada si se utilizan de manera correcta las librerías apropiadas para ello, pero si es necesario introducir dichos códigos para el reconocimiento de la manera propicia. En el caso de este proyecto se parte de una imagen con toda la información del documento agrupada en 21 códigos PDF417 en una imagen como la siguiente:

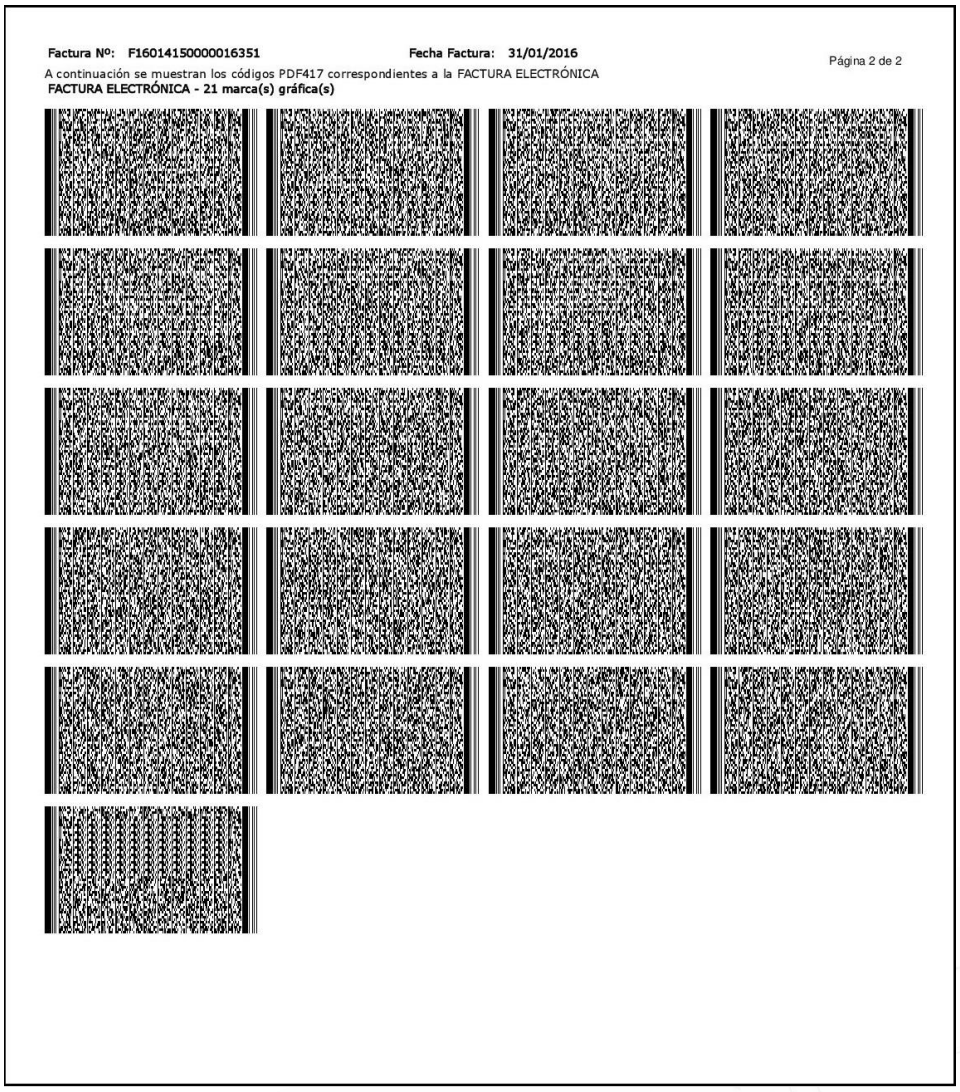


Figura 12. Imagen 21 códigos PDF417

Esta imagen de los códigos se podría reconocer completamente a la vez por un por un programa bastante robusto. En el caso de este proyecto lo que se hace es recortar cada código de barras y hacer un reconocimiento individual y más tarde se unirá toda la información reconocida de cada uno de ellos.

Dicho esto lo que hay que realizar es recortar cada uno de los rectángulos de código individual. Esto se puede hacer con código de programación de tratamiento de imágenes (que es lo que se usara en el programa final para seleccionar las regiones de interés) o recortando manualmente con alguna herramienta aparte externa al programa.

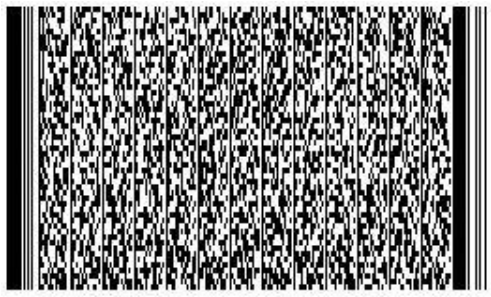


Figura 13. Código de barras PDF417 aislado

Por lo tanto se dispondrán de 21 códigos similares al de la imagen de la Figura 13 y se reconocerán uno a uno. Es importante que la imagen esté bien recortada, puesto que si hay mucho espaciado blanco en los bordes antes de ser el código como tal, puede que no se realice el reconocimiento. También es importante que la imagen tenga una calidad adecuada por lo que hay que llevar cuidado con las transformaciones de formato y tamaño que se realicen a la imagen original hasta llegar a cada código.

3.4.2 CÓDIGOS PDF417 – RECONOCIMIENTO

Una vez que ya se tienen cada uno de los códigos de manera individual se pasa al reconocimiento de estos y a la unión de toda la información. La manera de hacer esto es mediante código de programación "jugando" con bucles y cadenas de caracteres. Por ello es importante que cada uno de esos códigos individuales se haya guardado con el nombre adecuado y de manera ordenado. Los códigos tienen el orden de manera natural de lectura europea y se guardan como 1.jpg, 2.jpg, 3.jpg,……, 21.jpg.

El código completo se muestra al final del documento en la sección de código, pero a continuación se muestran algunas partes de interés.

```
const char* pszSettingFile = "C:\\Program Files
(x86)\\Dynamsoft\\Barcode Reader
6.0\\Templates\\default_settings.json";
const char* pszTemplateName = NULL;
char pszBuffer[512] = { 0 };
char pszImageFile[512] = { 0 };
char str[512] = "C:\\Users\\User\\Desktop\\IMAGENES PDF
TFG\\x.jpg";
char str1[512] = "C:\\Users\\User\\Desktop\\IMAGENES PDF
TFG\\xx.JPG";
char inicio[512] = { 0 };
char Cadena_tot[4096] = { 0 };
char Final[32768] = { 0 };
```

En el main, en la parte de declaración de variables declaradas, las cadenas "str" y "str1" deben llevar la dirección de localización de las imágenes de los códigos a reconocer. Además es de especial importancia el tamaño de las cadenas declaradas y posteriormente su llamada en funciones puesto que son muy habituales los errores por incompatibilidades de tamaño.

```
CBarcodeReader reader;
	reader.InitLicense("t0068MgAAAEKRJW/TAwoOhPqgdhWJ1k0dsRKPVALwWnHBn
	dwHPv1tVUAJWzblySqmMQytsZToZNovsDs40CLXdsKfnmPKG0A=");
	iRet = reader.LoadSettingsFromFile(pszSettingFile, szErrorMsg,
256);
	if (iRet != DBR_OK)
	{
		printf("Codigo erroneo: %d. mensaje de error: %s\n", iRet,
szErrorMsg);
		return -1;
	}
```

Esta parte es la de la licencia de lectura y puede cambiar dependiendo del momento de descarga del programa del que se inspira dicho código o de la versión de Dynamsoft Barcode Reader.

```
iIndex = 14;            //14 es para PDF417
pszTemplateName = GetTemplateName(iIndex);
if (pszTemplateName != NULL)
	break;
```

Esta parte de código sirve para decirle al programa de qué tipo de código de barras estamos realizando el reconocimiento; en este caso el número 14 se refiere a códigos PDF417.

```
ullTimeBegin = GetTickCount();
iRet = reader.DecodeFile(pszImageFile, pszTemplateName);
ullTimeEnd = GetTickCount();
```

En la segunda línea de esta parte de código es donde se realiza el reconocimiento propiamente dicho de los códigos de barras. La primera y tercera línea sirve para inicializar contadores y así ser capaces de ver cuánto tiempo tarda dicho reconocimiento.

```
STextResultArray *paryResult = NULL;
reader.GetAllTextResults(&paryResult);

if (paryResult->nResultsCount == 0)
{
	std::cout << std::endl << "No se ha encontrado el código. Tiempo
total: " << ((float)(ullTimeEnd - ullTimeBegin) / 1000) << "
segundos."<< std::endl;
	CBarcodeReader::FreeTextResults(&paryResult);
	continue;
                    }
segundos = ((float)(ullTimeEnd - ullTimeBegin) / 1000) + segundos;
if (pasos){
	std::cout << std::endl << "No se ha encontrado el código. Tiempo
total: " << ((float)(ullTimeEnd - ullTimeBegin) / 1000) << " segundos."
<< std::endl;
                    }
for (iIndex = 0; iIndex < paryResult->nResultsCount; iIndex++)
{
	if (pasos)
	{
		std::cout << std::endl << "Codigo de barras " << i <<
std::endl;
		std::cout << std::endl << "    Tipo " << paryResult-
>ppResults[iIndex]->pszBarcodeFormatString << std::endl;

		std::cout << std::endl << "Valor: " << paryResult-
>ppResults[iIndex]->pszBarcodeText << std::endl;
	}

strcpy_s(Cadena_tot, paryResult->ppResults[iIndex]->pszBarcodeText);
Cadena_tot[strlen(Cadena_tot)] = '\0';
Final[strlen(Final)] = '\0';
strcat_s(Final, Cadena_tot);
Final[strlen(Final)] = '\0';
```

En este último tramo de código de esta parte lo que se hace es crear una variable específica donde se guardan los datos del reconocimiento (“`STextResultArray *paryResult”)` y luego se muestran los datos (“pasos” es una variable creada que recibe la respuesta por teclado de si se quieren ir mostrando los datos). Además se ve cómo se va juntando toda la información reconocida en la cadena “Final” que es la que servirá posteriormente.

Cabe destacar, que el reconocimiento de estos códigos de barras de tipo PDF417 es bastante robusto. Esto se puede comprobar en que la información se reconoce de igual manera tanto si el código de barras está en su posición original como si esta rotado 180º. Además si el código puede ser reconocido si está levemente alterado (teóricamente hasta un 40% de sus píxeles).

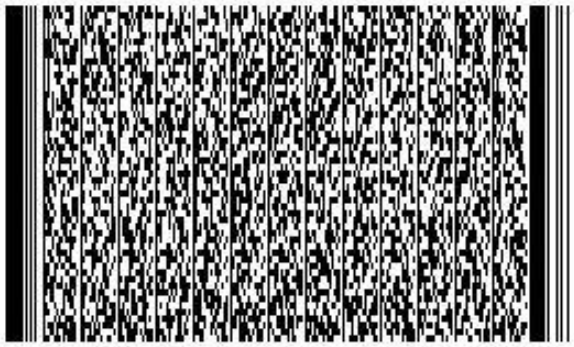


Figura 14. Código PDF417 original

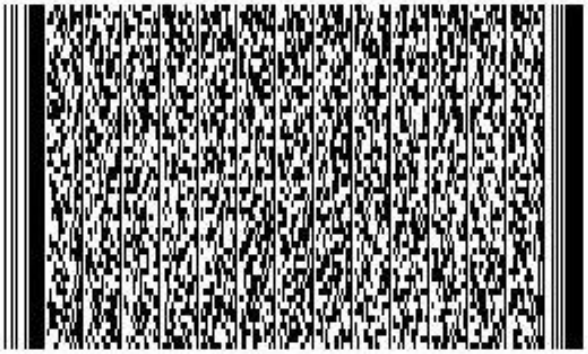


Figura 15. Código PDF417 girado 180º

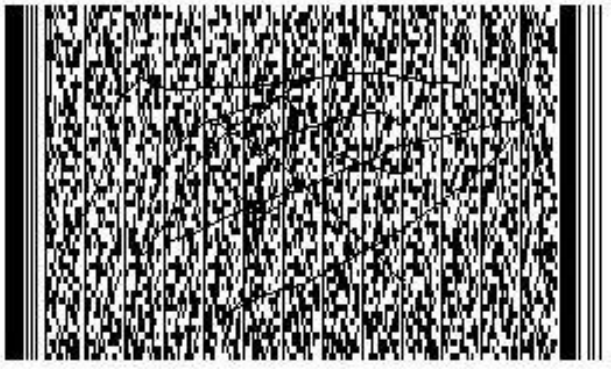


Figura 16. Código PDF417 rayado

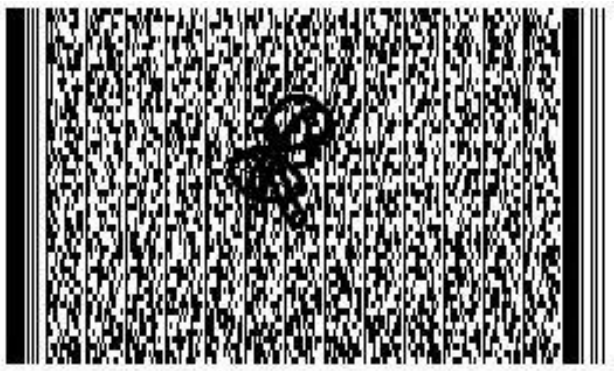


Figura 17. Código PDF417 estropeado

Los tres primeros códigos mostrados justamente encima, tienen el mismo reconocimiento de la información encriptada, siendo esto una muestra de la robustez de este tipo de códigos. , pero en el cuarto código el programa ya no es capaz del reconocimiento.

A continuación se muestra como sería el reconocimiento de uno de los códigos aislados (en este caso se trata del segundo):

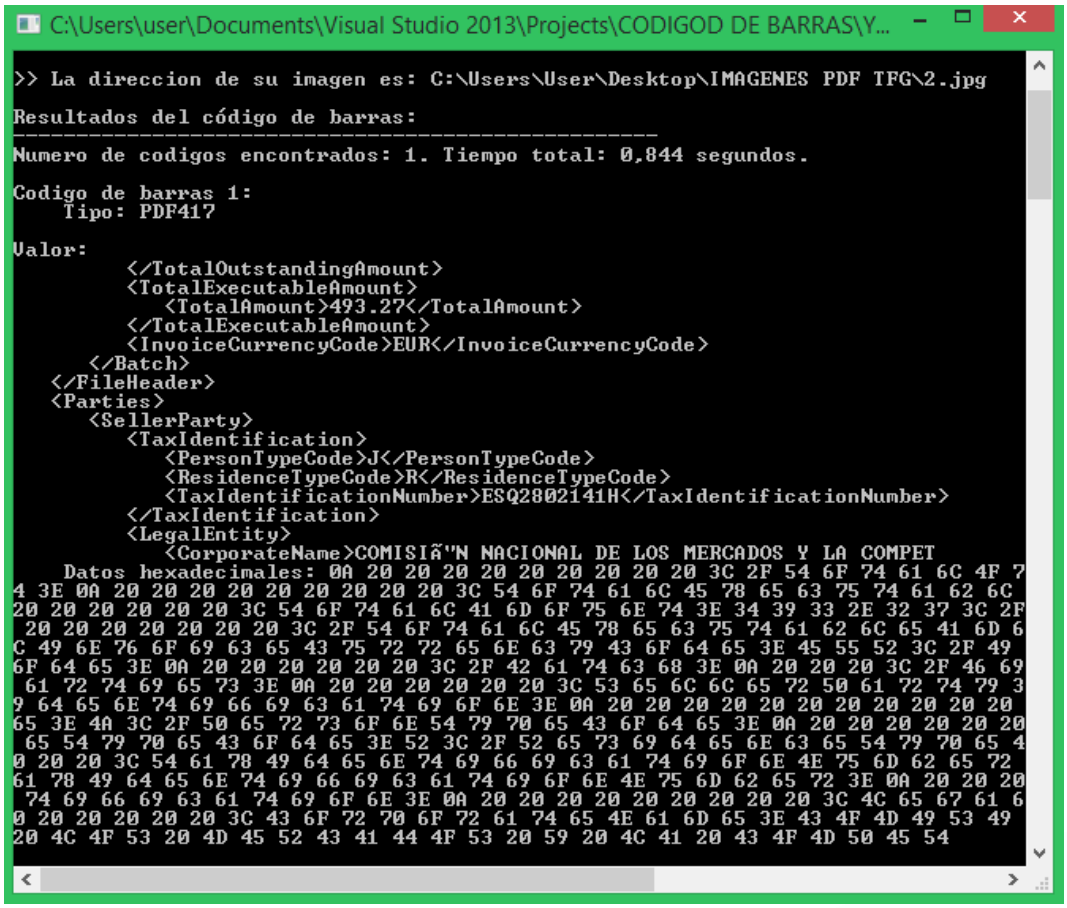


Figura 18. Resultado reconocimiento código PDF417 aislado

Nota: el reconocimiento de todos los códigos a la vez es lo mismo pero enlazando la información de cada uno de ellos.

Se puede ver como la información queda cortada en el final del texto reconocido pues deja la palabra "COMPETENCIA" truncada a "COMPET". El siguiente código reconocido empezará por "ENCIA" y al unirlos todos tendrá sentido la información completa del documento.

3.5 IMPLEMENTACIÓN DE PRIMITIVAS DEL LENGUAJE DE USUARIO Y DEL LENGUAJE PARA EL CARGADO DE IMÁGENES Y PRE-PROCESAMIENTO

Antes de comenzar los capítulos 3.5 y 3.6 en los que se trata del funcionamiento del programa final, hay que destacar dos cosas:

1. El programa final está dividido en varios ficheros tanto ".cpp" con el código propiamente dicho como ".h" con los inlcudes necesarios y declaración de funciones. Esto se hace debido a las múltiples ventajas de la programación modular para un programa extenso como es el caso de este proyecto.
2. El archivo más importante de dicho programa es el que se llama "ExtractData", el cual sería la primitiva que se llamaría desde un programa externo más grande en el que introduciendo los datos necesarios se conseguiría el reconocimiento del documento completo.

Dicho lo anterior, hay que mencionar que en esta parte del proyecto se tratarán las primitivas, con sus funciones, que tienen que ver con el título del apartado. A partir de este momento las partes que se expliquen corresponden ya al programa definitivo en el que se engloba todo lo necesario para el funcionamiento del programa.

3.5.1 LENGUAJE DE USUARIO

En el desarrollo de este proyecto se elaboró inicialmente una interfaz en Visual Basic para la comunicación con el usuario, pero debido a inconvenientes de este lenguaje como son la inflexibilidad, su lentitud para aplicaciones grandes y que sus ejecutables solo sirven para Windows, se optó por dejar de lado esa opción y

programar todo en C++ debido a sus principales ventajas de robustez, flexibilidad y eficiencia. Por este motivo no es fácil la creación de una interfaz de usuario, y la comunicación entre dicho usuario y el ordenador se realizar a través de la consola de comandos.

La principal necesidad del usuario para el reconocimiento es para que seleccione el archivo, el cual quiere reconocer y el tipo de documento del que se trata. En este caso como tenemos el ejemplo de una factura solo se dispone del tipo de documento factura, pero en el caso de que hubiera más tipos habría que añadir sus características como base de conocimiento en el archivo "ExtractData.h".

Por tanto lo primero que se hace al iniciar el programa (después de cargar el entrenamiento KNN para poder usarlos posteriormente), es preguntar al usuario esos dos datos claves para nuestro programa:

```
	while (1)
	{

		std::wcout << std::endl << "Paso 1: Introduce el nombre
de su imagen: " << std::endl;
		gets_s(nombreimagen, 512);	//guarda lo introducido en
pszBuffer
		iLen = strlen(nombreimagen);	//tamaño de la cadena
introducida
		nombreimagen[iLen] = '\0';

		while (1){
			std::wcout << std::endl << "Paso 1: Introduce el
modelo de tipo de documento: " << std::endl;
			std::wcout << std::endl << "Para factura pulse 1 "
<< std::endl;
			std::wcout << std::endl << "Para contrato pulse 2 "
<< std::endl;
			gets_s(nombremodelo, 512);	//guarda lo
introducido en pszBuffer
			iLen = strlen(nombremodelo);	//tamaño de la
cadena introducida
			numero = atoi(nombremodelo);

			if (numero == 1){
				break;
			}
			std::cout << std::endl << "Por favor introduce el
numero de un modelo registrado";
		}
```

Como se puede observar se pregunta por los datos de imagen y modelo y se guardan en las variables que más tarde se le pasarán a ExtractData. En este caso se puede observar que se ha hecho un bucle para que sea necesario que el modelo introducido sea una factura, pero en una aplicación más este bucle no tendría sentido, puesto que se podría seleccionar cualquier tipo de documento.

Aunque esto es la comunicación principal con el usuario, más tarde, conforme se va ejecutando el programa, se necesitan otras respuestas del usuario para que este funcione de una manera u otra. Estas son algunas como de que parte quiere que se realice el reconocimiento, si se quiere comprobar lo reconocido o si se quiere que se muestren los pasos de reconocimiento de los códigos de barras. Estas preguntas se engloban dentro del archivo ExtractData.cpp y se pueden comprobar en la sección de código del final del proyecto.

3.5.2 CARGADO DE IMÁGENES

El cargado de imágenes como se explica en el apartado 3.2 se hace de manera manual. La imagen que se desea cargar debe de estar en la misma carpeta que la del proyecto (esto se hace simplemente para simplificar y se le pueda llamar simplemente por su nombre y no haga falta poner toda la ruta de su localización) y se le debe llamar con su nombre completo (la terminación de formato de imagen también).

En el caso de este proyecto se pregunta inicialmente en el archivo Main.cpp por el nombre de la imagen, pero es en el ExtractData.cpp donde se realiza el cargado de la imagen.

```
cv::Mat imgOriginalScene;           // imagen original

imgOriginalScene = cv::imread(Imagen);         // abrir la imagen
deseada
       if (imgOriginalScene.empty()) {
// y si no se puede abrir
              std::cout << "error: imagen no leida \n\n";
// mostrar un mensaje de error
              _getch();
// esperar a pulsar una tecla para salir del programa
              return 0;
// salir
       }
```

Se crea una matriz en la que se guarda lo leído de "Imagen" (esta variable recibe el nombre de la imagen introducido al principio) y muestra un mensaje de error en caso de que lo haya.

Aunque lo que se pregunta inicialmente es la imagen del documento del cual se quiere obtener la información y es esta imagen la que se carga, a la hora de realizar el reconocimiento solo se pasa al programa que se encarga de reconocer la parte de la imagen la cual se quiere identificar.

En la primitiva ExtractData se dispone como base de conocimiento la localización de cada parte a reconocer en función del tipo de documento del que se trate; en este caso solo el caso de factura.

```
       switch (Modelo){
              //En caso de tener otros tipos de documentos las regiones
de interés serían otras en los distintos case
       case 1:
              cv::Rect Emisor(440, 545, 670, 75);
              imgROIEmisor = imgOriginalScene(Emisor);
              //cv::imshow("imgROIEmisor", imgROIEmisor);
              //cv::waitKey();
              cv::Rect Productor(1430, 550, 450, 47);
              imgROIProductor = imgOriginalScene(Productor);
              //......
              //el resto de datos que se quieran reconocer
              //......
       }
```

3.5.3 PRE-PROCESAMIENTO

En este apartado lo que se explica son las primitivas de pre-procesamiento del programa completo final. Siguen la idea que se trata en el capítulo 3.2 de pre-procesamiento de imágenes pero esta vez de una manera más extensa y específica para las necesidades del proyecto.

La parte de programa relacionada con este apartado se encuentra en el archivo preprocesamiento.cpp y como todo código se encuentra en la sección de código fuente.

Aunque en el apartado 3.2 se utiliza la imagen del documento completo como base de trabajo, en este apartado lo que se está pre-procesando son los recortes de la imagen original donde se encuentra la información de interés que se quiere reconocer. En todo este apartado se pondrá de ejemplo la imagen del nombre de la entidad emisora de la factura.

Figura 19. Entidad emisora original

A continuación se comentan las partes más destacadas.

Después de obtener la imagen en escala de grises de manera análoga a como se hacía en el apartado de "Pruebas aisladas de cargado de imágenes y pre-procesamiento", lo que se hace es maximizar el contraste para resaltar y remarcar las diferencias de luminosidad de las distintas partes de la imagen (ajusta la diferencia entre partes claras y oscuras).

```
cv::Mat maximizeContrast(cv::Mat &imgGrayscale) {
	cv::Mat imgTopHat;
	cv::Mat imgBlackHat;
	cv::Mat imgGrayscalePlusTopHat;
	cv::Mat imgGrayscalePlusTopHatMinusBlackHat;

	cv::Mat structuringElement =
cv::getStructuringElement(CV_SHAPE_RECT, cv::Size(3, 3));

	cv::morphologyEx(imgGrayscale, imgTopHat, CV_MOP_TOPHAT,
structuringElement);
	cv::morphologyEx(imgGrayscale, imgBlackHat, CV_MOP_BLACKHAT,
structuringElement);

	imgGrayscalePlusTopHat = imgGrayscale + imgTopHat;
	imgGrayscalePlusTopHatMinusBlackHat = imgGrayscalePlusTopHat -
imgBlackHat;

	return(imgGrayscalePlusTopHatMinusBlackHat);
}
```

En este caso la morfología que se usa a diferencia de en el apartado 3.2 que se usaba un cierre aquí se opta por un TopHat (diferencia entre original y apertura) y un BlackHat (diferencia entre cierre y original) y se aplican ambas a la imagen original, obteniendo con estas operaciones el mejor resultado.

Figura 20. Entidad emisora con máximo contraste

Nota: aunque se parezca bastante a la imagen original se puede observar la diferencia principalmente en el grosor de las letras. En este caso por ejemplo los huecos de las “A” y las “R” son más grandes, lo que facilita el reconocimiento.

Finalmente lo que se realiza antes de entrar en el reconocimiento, es hacer el emborronamiento gaussiano que sirve para difuminar la imagen y así quitar posible ruido de la imagen, y por ultimo mediante el método del valor umbral hacer resaltar las imágenes del fondo de la imagen para su mejor identificación.

```
cv::Mat imgBlurred;
cv::GaussianBlur(imgMaxContrastGrayscale, imgBlurred,
GAUSSIAN_SMOOTH_FILTER_SIZE, 0);          // emborronamiento gausiano
cv::adaptiveThreshold(imgBlurred,           imgThresh,           255.0,
CV_ADAPTIVE_THRESH_GAUSSIAN_C,                    CV_THRESH_BINARY_INV,
ADAPTIVE_THRESH_BLOCK_SIZE, ADAPTIVE_THRESH_WEIGHT);
```

Como se observa se trata de las mismas funciones de emborronamiento y método del valor umbral que se usan en la sección 3.2 y dan como resultado las siguientes dos imágenes.

Figura 21. Entidad emisora difuminada

Figura 22. Entidad emisora-Valor umbral

Esta última imagen es la que se intentaría identificar y estos pasos, como se ha dicho previamente, se hacen para cada información o dato a reconocer.

3.6 IMPLEMENTACIÓN DE PRIMITIVAS DE PROCESAMIENTO Y SALIDA

En este apartado se toma como idea de partida lo tratado en los capítulos 3.3 "Pruebas aisladas de procesamiento de imágenes" y 3.4 "Pruebas aisladas de procesamiento de códigos de barras" y por ello aparece la necesidad de explicarlos por separado.

Por otro lado se explicará la manera en la que se obtiene la información y las distintas posibilidades que hay a la hora de finalizar el programa para que la identificación del documento sea la adecuada en cualquiera de los casos estudiados que posteriormente se comentan.

3.6.1 PROCESAMIENTO DE IMÁGENES

Esta sección al igual que el capítulo 3.5 se basa principalmente en el método KNN. El entrenamiento no hace falta volver a hacerlo otra vez, ya que los caracteres representados en la Figura 9 son los caracteres de interés también para el programa final y cuyos datos ya se guardaron una vez. Lo único que hay que hacer es copiar los archivos classifications.xml e images.xml, los cuales tienen las estructuras de datos necesarios para el reconocimiento y pegarlos en la carpeta de trabajo del proyecto final.

Nada más empezar el programa completo lo primero que se hace es cargar la información de esos datos del entrenamiento en dos matrices para su posterior uso a la hora del reconocimiento, y así si hay algún problema con este paso mostrar un mensaje de error puesto que si este cargado no se logra con éxito el resto de pasos no tienen sentido.

Tras el cargado del entrenamiento KNN lo siguiente es la llamada al reconocimiento.

El reconocimiento se hace de manera análoga a cómo se hace en el apartado 3.3.3. Las principales diferencias son:

- El pre-procesamiento de la imagen se hace de manera externa con la primitiva explicada anteriormente en la sección 3.5.3.
- Se enriquece el código tanto estéticamente como añadiendo líneas de código que ayudan al reconocimiento como que el reconocimiento de los caracteres no solo se haga de izquierda a derecha, sino que también separe en caso de que haya dos renglones, con la parte que se muestra a continuación.

```
if (vectorOfMatchingChars[i].intCenterY < (imgOriginal.rows * 0.5))
{
		strChars1 = strChars1 + char(int(fltCurrentChar));
	}
else{
		strChars2 = strChars2 + char(int(fltCurrentChar));
	}
strChars = strChars1 + strChars2;
```

- Se añaden condiciones para que los caracteres sean reconocidos como válidos y ya no es simplemente una comparación de tamaño del carácter.

```
bool checkIfPossibleChar(PossibleChar &possibleChar) {
	// esta función es solo una pequeña comprobación que mira las
dimensiones del contorno para ver si puede ser un caracter
	// no hace comparación entre caracteres
	if (possibleChar.boundingRect.area() > MIN_PIXEL_AREA &&
		possibleChar.boundingRect.width > MIN_PIXEL_WIDTH &&
possibleChar.boundingRect.height > MIN_PIXEL_HEIGHT &&
		MIN_ASPECT_RATIO < possibleChar.dblAspectRatio &&
possibleChar.dblAspectRatio < MAX_ASPECT_RATIO) {
		return(true);
	}
	else {
		return(false);
	}
}
```

En este caso se ponen unas condiciones de tamaño mínimo de área, anchura y altura del píxel, así como una relación de forma entre altura y anchura.

Nota: todos los valores de los parámetros hay que ajustarlos hasta que el reconocimiento sea el mejor posible.

Se podrían añadir otras condiciones como la distancia o el ángulo entre caracteres pero después de probarlos en nuestro proyecto no se ha encontrado utilidad en ellas y se ha optado por obviarla.

3.6.2 PROCESAMIENTO DE CÓDIGOS DE BARRAS

De igual manera que se ha explicado en el apartado 3.5, el procesamiento de los códigos de barras es clave principalmente para la comprobación y confirmación de que la información reconocida es la correcta, puesto que en dichos códigos PDF417 se encuentra encriptada los datos del documento.

En el caso del programa final una vez que se manda hacer el reconocimiento de los códigos, se hace enteramente de manera continua y la condición de mostrar o no los pasos, se pregunta externamente a la primitiva y se le pasa como un parámetro quedando la definición de esta de la siguiente manera:

```
std::string comprueba(char letra, std::string parecido, int pasos);
```

Teniendo en cuenta esta declaración, el parámetro "letra" se encarga de pasar de que parte del documento se está realizando el reconocimiento (entidad emisora, entidad productora…), el parámetro "parecido" se encarga de enviar como una cadena lo que se ha obtenido del reconocimiento con el procesamiento de imágenes y el entero "pasos" sirve para indicar si se quieren mostrar los pasos del reconocimiento. Se devolverá finalmente una cadena que contiene la información del dato del documento que se está reconociendo, pero de manera apropiada (en

los códigos está la información correcta). En el programa final el sentido de reconocer los códigos no es obtener la información porque sí, sino ser capaz de ver que parte del documento de la imagen se está intentando reconocer y devolver ese dato que estará en alguno de los códigos.

Para ser capaz de devolver la información por la que se está preguntando de entre toda la que hay en los códigos, primero hay que estudiar dicha información para a través de cadenas de caracteres, crear los bucles necesarios con la información de conocimiento base necesaria. A continuación se muestra una imagen de un fragmento de la información total de los códigos para explicar lo tratado.

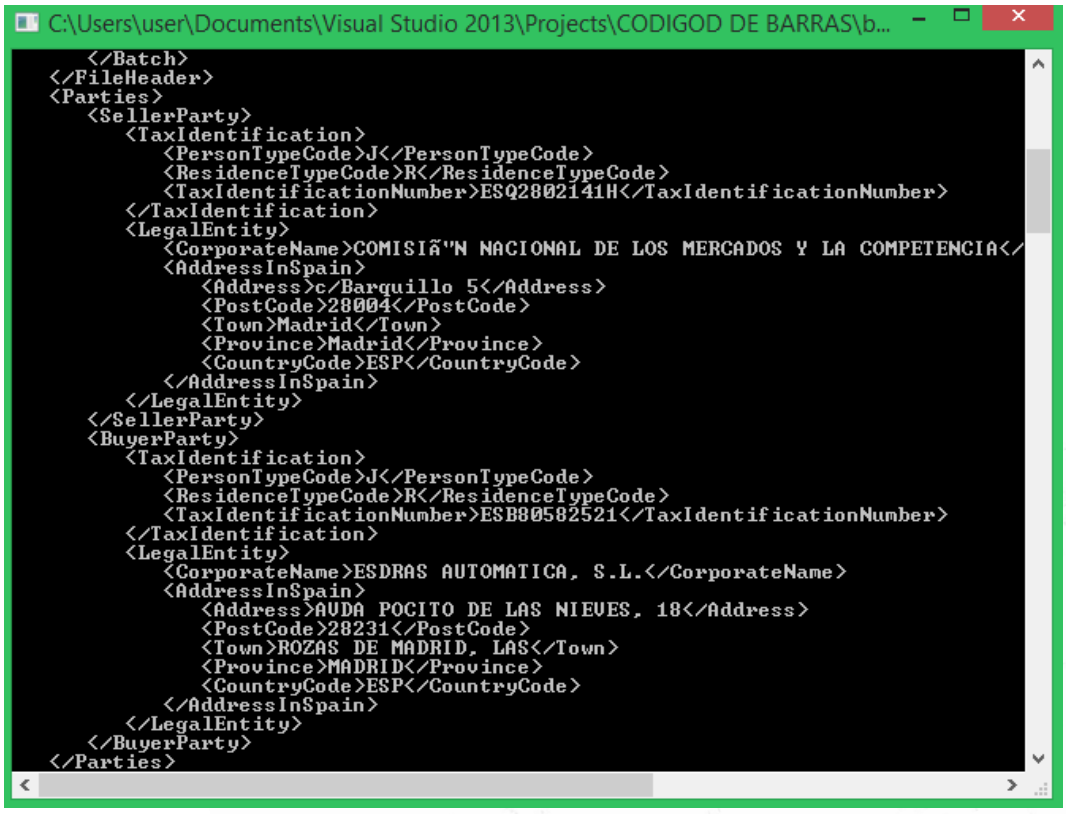


Figura 23. Fragmento del reconocimiento de los códigos PDF417

Como se puede observar, tanto la información de la entidad emisora ("COMISION NACIONAL DE LOS MERCADOS Y LA COMPETENCIA") como la de la entidad productora ("ESDRAS AUTOMATICA, S.L."), que se puede observar en la Figura 5 de la imagen original, se encuentra, en este caso de la información leída de los códigos, tras el identificador "CorporateName". Para

distinguir donde buscar la información dependiendo cuál de los dos datos se necesite, hay que buscar cual es la primera diferencia de localización en la cadena, observándose que la entidad emisora está dentro de la sección "SellerParty" mientras que la productora en "BuyerParty".

Esto es lo que hay que saber para obtener la información adecuada manejando para ello los bucles con cadenas de caracteres como el que se explica a continuación (para diferenciar estos dos datos de interés del ejemplo).

```
const char* ObtenerQuiero(char quiero)
{
	switch (quiero)
	{
	case 'e':
		return "SellerParty";
	case 'p':
		return "BuyerParty";
	default:
		return NULL;
	}
}
```

Primero cabe mencionar que dependiendo del dato del que se está realizando la comprobación para un mejor reconocimiento, el cual se introduce por pantalla y son los distintos casos del switch, se dispondrá de una palabra clave u otra, y en la sección de código que se acaba de presentar se ve el caso para este ejemplo que se está tratando.

```
strcpy_s(copia, ObtenerQuiero(letra));

char nombre[1024] = "CorporateName";

for (l = 0; l < strlen(Final); l++)
	if (Final[l] == copia[0])
	{
		if (strlen(Final) - l < strlen(copia) || fin == 1)
			return "error";

		for (j = 0; j < strlen(copia); j++)
			if (Final[l + j] != copia[j] || fin == 1)
				break;

		if (j == strlen(copia)){
			correcto = 1;
```

```
                    if (correcto)
                    {
                           for (l = (l + j); l < strlen(Final); l++)
                                  if (Final[l] == nombre[0])
                                  {
                                         if (strlen(Final) - l <
strlen(nombre))
                                                return "error";


                                         for (j = 0; j <
                                  strlen(nombre); j++)
                                                if (Final[l + j] !=
nombre[j] || fin == 1)
                                         break;
                                         if (j == strlen(nombre)){
                                         correcto = 1;

                                                      if (correcto)
                                                      {
                                                             while
(Final[l + j + 1] != '<')
                                                             {

       cadena[k] = Final[l + j + 1];

       k++;

       l++;

                                                             }
                                                             fin = 1;
                                                      }
                                                }
                                         }
                           }
                    }
             }
```

Es una sección difícil de comprender a primera vista pero que sirve para identificar la diferencia entre los dos tipos de entidades. Es muy importante el uso del debugger para ir viendo de qué manera se recorre la cadena completa y se va haciendo la comparación. En los bucles se va comprobando la información total del documento y en función de lo que se le haya mandado reconocer devolverá una información u otra.

La declaración de las variables y el código completo de esta parte se encuentran en la sección de código fuente.

3.6.3 PRIMITIVAS DE SALIDA

En esta sección se explica la manera que tiene el programa de mostrarnos finalmente aquello que nos interesa, que no es más que el reconocimiento de los distintas partes del documento para una posible clasificación posterior.

Como se explica en el apartado 3.5.1, no se desarrolla una interfaz en este proyecto. Además de las razones dadas en el apartado recientemente mencionado, para la salida tampoco hay motivo superior para su realización, puesto que se trata de un proyecto basado en un programa de clasificación y por tanto la manera de visualización no es tan importante como se podría pensar.

Por tanto al no haber interfaz específica, la información se mostrará por pantalla a través de la consola de comandos.

Como ya se ha mencionado en otras ocasiones cabe destacar que el fichero ExtractData es el más importante de este proyecto y al llamarlo es en él donde se muestran los resultados.

```
	std::cout << std::endl << "De que quiere realizar el
reconocimiento? " << std::endl;
	std::cout << std::endl << "Para todo pulse ---------------- t" <<
std::endl;
	std::cout << std::endl << "Para entidad emisora pulse ----- e" <<
std::endl;
	std::cout << std::endl << "Para entidad productora pulse -- p" <<
std::endl;
	gets_s(reconoce, 256);	//guarda lo introducido en reconoce

	if (strlen(reconoce) == 1 && (reconoce[0] == 'e' || reconoce[0]
== 'E' || reconoce[0] == 't')) // salir si se pulsa q
	{
		datEmisor = recognizeCharsInPlate(imgROIEmisor);
		std::cout << std::endl << "El nombre de la entidad emisora
es = " << datEmisor << std::endl;
	}

	if (strlen(reconoce) == 1 && (reconoce[0] == 'p' || reconoce[0]
== 'P' || reconoce[0] == 't')) // salir si se pulsa q
	{
		datProductor = recognizeCharsInPlate(imgROIProductor);
		std::cout << std::endl << "El nombre de la entidad
productora es = " << datProductor << std::endl;
	}
```

```
_getch();

	std::cout << std::endl << "Quiere comprobar si el reconocimiento
ha sido correcto con toda la informacion de los codigos de barras" <<
std::endl;
	std::cout << std::endl << "s=si / n=no" << std::endl;
	gets_s(quiero, 256);	//guarda lo introducido en quiero
	if (strlen(quiero) == 1 && (quiero[0] == 's' || quiero[0] ==
's')){ // salir si se pulsa q
		std::cout << std::endl << "Quieres ver paso a paso el
reconocimiento de cada codigo?" << std::endl;
		std::cout << std::endl << "s=si / n=no" << std::endl;
		gets_s(quiero, 256);
		if (strlen(quiero) == 1 && (quiero[0] == 's' || quiero[0]
== 's'))
			pasos = 1;
		if (strlen(reconoce) == 1 && (reconoce[0] == 'e' ||
reconoce[0] == 'E' || reconoce[0] == 't')){
			strcpy_s(correccion, (comprueba('e', datEmisor,
pasos)).c_str());
			std::cout << std::endl << "Entidad emisora
correcta: " << correccion << std::endl;
		}

		if (strlen(reconoce) == 1 && (reconoce[0] == 'p' ||
reconoce[0] == 'P' || reconoce[0] == 't'))
		{
			strcpy_s(correccion, (comprueba('p', datProductor,
pasos)).c_str());
			std::cout << std::endl << "Entidad productora
correcta: " << correccion << std::endl;
		}
	}
```

En esta sección de código es donde se muestra la información reconocida de la imagen y su correspondiente correcta de los códigos. En el capítulo siguiente de "Integración y pruebas" se puede ver cómo queda finalmente.

3.7 INTEGRACIÓN Y PRUEBAS

Una vez comentadas las distintas partes del proyecto, este capítulo sirve para explicar la manera en que dichas partes son llamadas y la relación que existe entre ellas, a la vez que se muestran los resultados de las pruebas de ejecución del programa final.

3.7.1 INTEGRACIÓN

Para entender el funcionamiento de programa completo, primero hay que comprender la manera en que todos los archivos, primitivas y funciones se relacionan entre sí y que tareas realizan, y de esta manera ser capaces de seguir el orden de ejecución.

Primero hay que aclarar que el programa está divido en seis ficheros ".cpp" con sus respectivos ".h" como se comenta al inicio del capítulo 3.5 y sus nombres son: Main, ExtractData, DetectChars, Preprocesamiento, PossibleChar y Correspondencia.

El cometido de cada uno de estos ficheros es el siguiente:

Nota: entre paréntesis se muestran los archivos con los que tiene relación directa.

- Main (ExtractData y DetectChars):
    - Llamada al cargado del entrenamiento KNN y comprobación de que se ha realizado correctamente.
    - Comunicación con el usuario para preguntar acerca del archivo a reconocer y el tipo de modelo.
    - Llamada a la primitiva ExtractData.
    - Finalización de la ejecución del programa.

- ExtractData (DetectChars y Correspondencia):

  Antes de mencionar la funcionalidad de este archivo, hay que comentar este archivo simula la primitiva que se llamaría desde un programa de más alta jerarquía. Por tanto tiene unas bases de conocimiento como son, por ejemplo, los recortes con la región de interés comentados en la sección 3.5.2 o los nombres posibles de las distintas entidades y así hacer también una comparación entre lo reconocido con el procesamiento de imágenes, y su parecido con esos posibles nombres.

  Dicho esto, lo que contiene, o de lo que se encarga el fichero con este nombre es lo siguiente:

  - En él se encuentran las bases de conocimiento de las regiones de interés y los posibles nombres de la información que tiene interés su reconocimiento.
  - Comunicación con el usuario para saber tanto de que parte se quiere hacer el reconocimiento como si se quieren hacer las comprobaciones con los códigos de barras PDF417 o con los posibles nombres.
  - Exponer por la consola de comandos los distintos resultados obtenidos.

- DetectChars (PossibleChar y Pre-procesamiento):
  - Cargado del entrenamiento KNN: se vuelcan los archivos ".xml" con las estructuras de datos a las matrices para su posterior uso.
  - Llamada al pre-procesamiento de la imagen.
  - Llamada a la comprobación de la validez de los caracteres.
  - Reconocimiento de los caracteres de la región de la imagen a identificar.
- Pre-procesamiento:
  - Conversión a escala de grises.
  - Maximizar el contraste.

- Emborronamiento gaussiano.
- Método del valor umbral.

- PossibleChar:
  - Creación de la clase PossibleChar.
  - Comprobación de que los contornos a valorar como caracteres cumple con los requisitos de carácter posible.
- Correspondencia:
  - Reconocimiento de la información de los códigos PDF417.
  - Identificación del dato que se está reconociendo en la imagen dentro de toda la información de los códigos.
  - Comparación entre la información obtenida con el procesamiento de imágenes y su mejor parecido entre los posibles nombres del dato en cuestión.

3.7.2 PRUEBAS DE LAS DISTINTAS ALTERNATIVAS

En este apartado se explican los resultados de las distintas pruebas necesarias para la comprensión del proyecto.

Para este cometido se utiliza solamente los datos de las entidades emisora y productora como datos de interés para simplificar su entendimiento y su presentación por pantalla. El programa completo tendrá todos los datos del documento, las distintas opciones, lo cual se consigue simplemente añadiendo líneas de código para variables y regiones de interés principalmente.

Como se ha visto en los anteriores apartados, se han ido realizando todo los procedimientos necesarios tanto de tratamiento de imágenes, entrenamiento KNN, lectura de códigos PDF417 etc., para conseguir la información de la manera más precisa posible. Una vez realizado todo eso, falta obtener esa información.

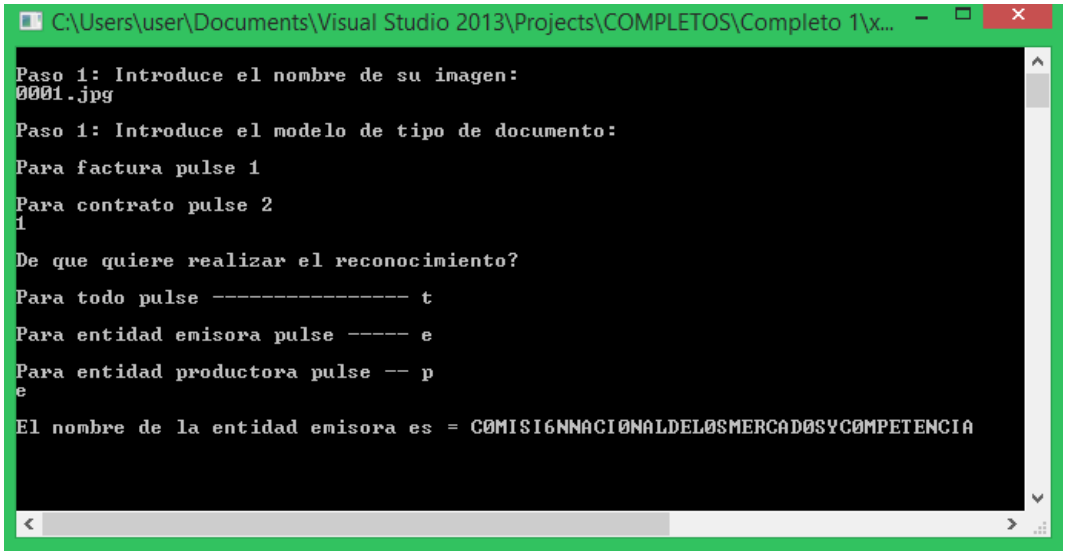


Figura 24. Reconocimiento de la entidad emisora sin comprobaciones

Como se observa se piden los datos de la imagen, el tipo de modelo y los datos por los que se preguntan, en este caso la entidad emisora. Además se aprecia que el mejor reconocimiento posible simplemente con el procesamiento de imágenes está, aún, un poco lejano de la información correcta.

Tras esto se puede comprobar, si se desea, cuál es la información correcta que se encuentra en los códigos de barras como se muestra en la siguiente imagen.

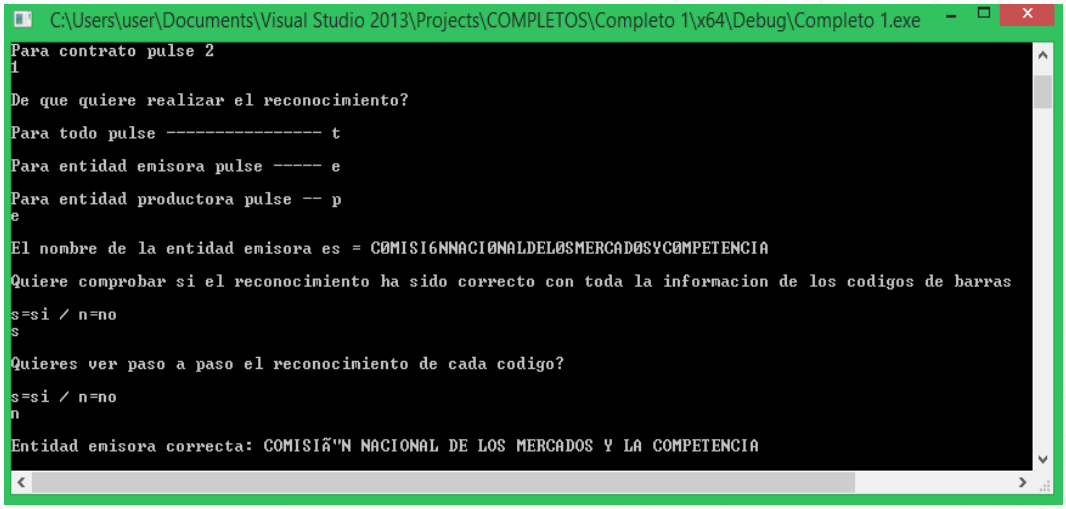


Figura 25. Reconocimiento de la entidad emisora comprobando con los códigos PDF417

Por últimos se pregunta al usuario si quiere realizar la comprobación con la información de los posibles nombres guardados de ese dato (en este caso de las entidades emisoras).

El resultado es el siguiente.

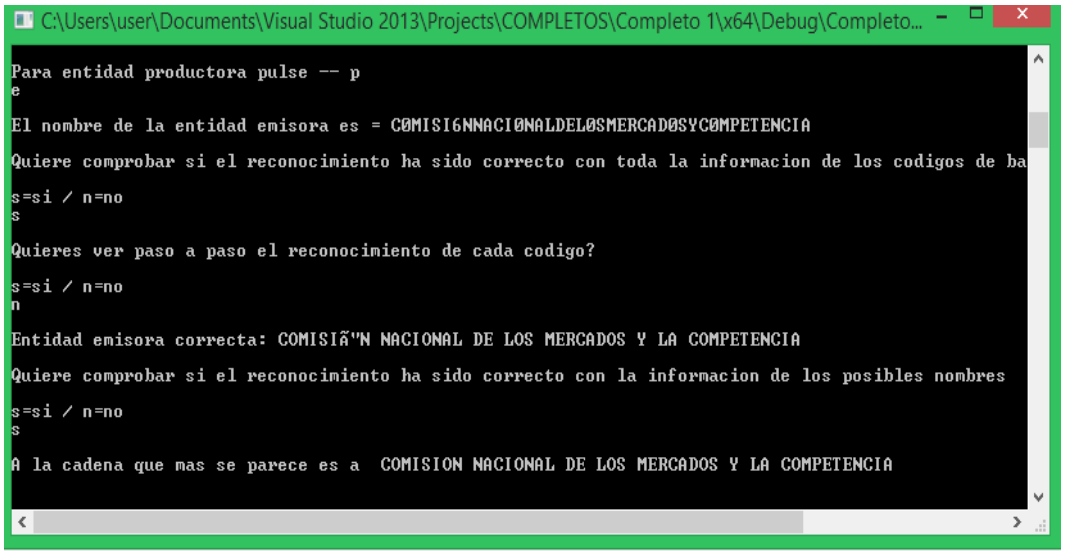


Figura 26. Reconocimiento de la entidad emisora comprobando los nombres posibles

Capítulo 4 CONCLUSIONES

Una vez acabado con el desarrollo del proyecto y haber analizado tanto el funcionamiento de este como sus resultados, se obtiene como consecuencia una seria de conclusiones que merece la pena comentar.

El objetivo general de conseguir desarrollar un programa para el reconocimiento de documentos de una manera computacional automática se ha conseguido de manera propicia y aunque se han encontrado inconvenientes durante este proceso se han intentado solventar de la mejor manera posible.

En primer lugar, observando los objetivos comentados al inicio de este documento, se ve como estos han sido alcanzados satisfactoriamente, en mayor o menor medida, como se explica a lo largo del capítulo 3.

En una primera instancia se pretendía conseguir el reconocimiento del documento, únicamente con las técnicas de visión artificial para procesamiento de imágenes, y tras analizar los resultados de esta fase, se ve como es excesivamente complicado conseguir una identificación de la información de manera perfecta, tanto debido a problemas de la imagen en reconocimiento (la cercanía de caracteres o la falta de calidad) como a lo arduo y complejo que resulta la realización de un programa lo suficientemente robusto para llevar a cabo la tarea (se necesitaría un entrenamiento con muchos más ejemplos, se tendrían que aplicar técnicas de inteligencia computacional etc.) que no está en el alcance de este proyecto. Aun así los resultados, aunque no perfectos, son razonablemente buenos e interpretables para poder realizar otras técnicas sobre ellos y mejor el reconocimiento.

Puesto que la idea es que este programa pueda tener una implementación real, el problema con el procesamiento de imágenes lleva a buscar nuevas soluciones. El reconocimiento de los códigos PDF417 es la herramienta que se escoge para mejor el programa y ayudarlo a que funcione de manera eficiente. En el caso los

códigos utilizados en este proyecto, estos contienen toda la información del documento y por tanto la búsqueda de la información de interés se hace directamente en ellos, pero también podría ser que contuviesen solamente alguna clase de información como la localización del recuadro dónde realizar el reconocimiento que en este caso se introduce en línea de código. En definitiva los códigos de barras tienen información adicional que ayudan al reconocimiento.

Además se utilizan otro tipo de métodos para aumentar la robustez del reconocimiento, como por ejemplo la identificación de la información reconocida con el procesamiento de imágenes con la información posible registrada sobre ese dato (se intenta simular eventos reales que podrían darse).

Se puede decir, por tanto, que con la unificación de todos estos procedimientos se consigue un programa lo suficientemente robusto y eficaz para su uso e implementación real.

Por último, pero no por ello menos importante, además de las conclusiones de ámbito técnico o tecnológico, merece la pena señalar como a lo largo de todo el proceso de realización del proyecto, así como en el estudio e investigación paralela, se han ido adquiriendo los conocimientos de programación, visión artificial, tratamiento de imágenes y códigos de barras etc., necesarios para la ejecución apropiada de este proyecto.

Capítulo 5 FUTUROS DESARROLLOS

El futuro más visible relacionado con este proyecto es la integración dentro de un programa mayor y su implementación en forma de aplicación (lo que llevaría consigo entre otras cosas un cambio de forma y la creación de una interfaz adecuada) y así tener un manejo más cómodo lo que aumente la utilidad de este proyecto. Una opción también es crear dicha aplicación para ser usada en dispositivos móviles, y de esta manera integrar los entornos de programación para dicho tipo de aplicaciones en el proyecto.

Otra posibilidad para enriquecer el desarrollo de este proyecto seria la implementación de procesos similares al reconocimiento óptico de caracteres (OCR), como por ejemplo, el reconocimiento óptico de marcas (OMR) o el reconocimiento inteligente de caracteres (IMR) y así crear un programa con mucha mayor funcionalidad.

Otra salida o continuación del trabajo aquí desarrollado sería la atribución de inteligencia a la aplicación. En la actualidad los términos de “Big Data” y “Machine Learning” están presentes en nuestro día a día en todos los ámbitos. La integración de ellos en este proyecto para que sea capaz de predecir o intuir qué se está reconociendo, es uno de los futuros trabajos que lo harían más eficaz y fiable.

BIBLIOGRAFÍA

[1] Fuente López, E. 2012. *Visión artificial industrial: procesamiento de imágenes para inspección automática y robótica*. Valladolid, España. Universidad de Valladolid, Secretariado de Publicaciones e Intercambio Editorial.

[2] Ordóñez L., J.P. 2009. Reconocimiento óptico de caracteres (OCR) con redes neuronales. Disponible en:

https://jpordonez.files.wordpress.com/2009/06/estado-del-arte.pdf

[3] Muñoz Manso, R. 2014, julio. Sistema de visión artificial para la detección y lectura de matrículas. Universidad de Valladolid, Trabajo fin de grado. Disponible en:

https://uvadoc.uva.es/bitstream/10324/11848/1/TFG-P-165.pdf

[4] Navarro Santapau, J. 2013, enero. Software de adquisición de imágenes y reconocimiento óptico de caracteres para Android. Universitat Oberta de Catalunya, Trabajo fin de máster. Disponible en:

http://openaccess.uoc.edu/webapps/o2/bitstream/10609/18691/9/jaimenavarrosantapauTFM0113memoria.pdf

[5] Arce Arroyo, A. 2016. Visión artificial de documentos. Escuela Técnica Superior de Ingeniería (ICAI), Trabajo fin de grado. Disponible en:

https://www.iit.comillas.edu/pfc/resumenes/578d1c4928c96.pdf

[6] Gupta, V. 2017, mayo. Color spaces in OpenCV (C++/Python). Disponible en:

https://www.learnopencv.com/color-spaces-in-opencv-cpp-python/

[7] Learn OpenCV. S.f. Disponible en:

https://www.tutorialspoint.com/opencv/index.htm

[8] Romero Galey, F.J. 2006, septiembre. Integración de sistema de visión artificial y robot en aplicación tipo pick&place. Escuela Superior de Ingenieros de Sevilla, Trabajo fin de carrera. Disponible en:

http://bibing.us.es/proyectos/abreproy/50031/fichero/Volumen+unico%252FPFC+Integracion+de+sistema+de+vision+artificial+y+robot+en+aplicacion+tipo+pick%26place+Sep2006.pdf

[9] Clarkk. 2016. Detect text in images with opencv. Foro. Disponible en: https://stackoverflow.com/questions/34415815/detect-text-in-images-with-opencv/34416601#34416601

[10] Morphological transformations. Última edición en 2014, noviembre. Disponible en: https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_imgproc/py_morphological_ops/py_morphological_ops.html

[11] Clasificación de imágenes: ¿cómo reconocer el contenido de una imagen? S.f. Universitat Autònoma de Barcelona. Curso. Disponible en: https://es.coursera.org/lecture/clasificacion-imagenes/k-nn-clasificacion-por-vecino-mas-cercano-QyKBk

[12] Aler Mur, R. S.f. Clasificadores KNN-I. Universidad Carlos III, Madrid. Disponible en: http://ocw.uc3m.es/ingenieria-informatica/analisis-de-datos/transparencias/KNNyPrototipos.pdf

[13] OpenCV documentation. Último acceso en 2018, abril. Disponible en: https://docs.opencv.org/2.4/index.html

[14] Inza, J. 2006. PDF-417. Qué es y para qué sirve. Disponible en: https://inza.wordpress.com/2006/09/14/pdf-417-que-es-y-para-que-sirve/

[15] Atyachin. 2012. ZBar vs. zxing - QR recognition comparison. Foro. Disponible en: https://stackoverflow.com/questions/8933033/zbar-vs-zxing-qr-recognition-comparison

[16] Dahms C. [Chris Dahms]. 2016, enero, 1. OpenCV 3 Windows 10 Installation Tutorial - Part 1 - C++. Vídeo. Disponible en: https://www.youtube.com/watch?v=7SM5OD2pZKY

[17] Dahms C. [Chris Dahms]. 2016, enero, 8. OpenCV 3 KNN Character Recognition C++. Vídeo. Disponible en: https://www.youtube.com/watch?v=CK0OCeCN9zg&t=17s

[18] Dahms C. [Chris Dahms]. 2016, enero, 9. OpenCV 3 License Plate Recognition C++ full source code. Vídeo. Disponible en: https://www.youtube.com/watch?v=euG7-o9oPKg&t=5s

[19] Computer vision. S.f. En Wikipedia. Recuperado en Noviembre de 2017. Disponible en: https://en.wikipedia.org/wiki/Computer_vision

[20] Optical carácter recognition. S.f. En Wikipedia. Recuperado en Diciembre de 2017. Disponible en: http://en.wikipedia.org/wiki/Optical_character_recognition

[21] Comparison of integrated development environments. S.f. En Wikipedia. Recuperado en febrero de 2018. Disponible en: https://en.wikipedia.org/wiki/Comparison_of_integrated_development_environments

[22] PDF417. S.f. En Wikipedia. Recuperado en mayo de 2018. Disponible en: https://es.wikipedia.org/wiki/PDF417

Parte II PRESUPUESTO

Capítulo 1 PRESUPUESTO

En este documento se analiza la parte económica del proyecto desarrollado.

Para ello se contabilizarán todos los recursos utilizados para la consecución de la ejecución final. Además se realiza una división de costes para cada uno de esos elementos necesarios.

Hay que tener en cuenta que el proyecto se supone desarrollado por un ingeniero técnico industrial y en un ámbito laboral.

1.1 COSTES DIRECTOS

En este apartado se comentan y analizan todos los elementos que, de una manera directa, son necesarios e imprescindibles para el desarrollo del proyecto. En este proyecto se hace el análisis de manera diferenciada entre "Herramientas y software" y "Mano de obra" que son los elementos que están claramente vinculados con el programa realizado.

1.1.1 MANO DE OBRA

A la hora de calcular cuánto es lo que cuestan las horas de trabajo dedicadas para la realización del proyecto, es importante hacer un estudio y desglose tanto de las horas trabajadas durante la realización del mismo como del precio de cada una de esas horas en función de la tarea que se estuviera realizando en dicho momento.

Horas dedicadas

Primero se verán cuantas son las horas dedicadas.

Se distinguen tres épocas bien diferenciadas: Octubre-Diciembre, Enero-Marzo, Abril-Junio.

- Octubre - Diciembre: en esta época hay que tener en cuenta que no se dedican horas los fines de semana, que hay periodo de exámenes y que hay varios días festivos. Además como se compagina con asignaturas las horas dedicadas son de 2h/día aproximadamente.

Días disponibles	90 días
Fines de semana	- 25 días
Época de parciales y finales	- 10 días
Festivos	- 5 días
Días totales dedicados	50 días
Horas por día dedicado	2 horas/día
Horas totales	100 horas

Tabla 1. Horas dedicadas entre octubre y diciembre

- Enero – Marzo: en esta etapa de desarrollo cabe destacar que si se trabaja los sábados pero hay más días festivos. Por otro lado las horas dedicadas asciende a tres horas al día.

Días disponibles	90 días
Fines de semana	- 13 días
Trabajos y entregas	- 5 días
Festivos	- 8 días
Días totales dedicados	64 días
Horas por día dedicado	3 horas/día
Horas totales	224 horas

Tabla 2. Horas dedicadas entre enero y marzo

- Abril – Junio: en este período se aumentan bastante las horas dedicadas al proyecto y se impulsa su desarrollo considerablemente. Las horas dedicadas don las siguientes.

Días disponibles	90 días
Período de exámenes	- 7 días
Festivos	- 3 días
Días totales dedicados	80 días
Horas por día dedicado	6 horas/día
Horas totales	480 horas

Tabla 3. Horas dedicadas entre abril y junio

Juntando los tres períodos se obtiene que las horas totales dedicadas del proyecto son aproximadamente 804 horas (se usaran 800 para trabajar)

A continuación se muestra las horas dedicadas a cada una de las tareas.

Tarea		Horas por tarea		Total horas
Investigación	Inicial	75	200	800
	Tratamiento imágenes	70		
	Códigos de barras	55		
Preparación del entorno	V.S. y OpenCV	30	80	
	Dynamsoft Barcode	50		
Desarrollo del programa	Efectivo	270	320	
	Inefectivo	50		
Pruebas y depuración			100	
Redacción			100	

Tabla 4. Horas dedicadas a cada tarea

Finalmente teniendo en cuenta el coste por hora dedicada a cada tarea se obtiene.

Tarea	Horas de tarea	Coste hora de tarea (€/horas)	Coste total de tarea (€)
Investigación	200	40	8000
Preparación del entorno	80	55	4400
Desarrollo del programa	320	55	17600
Pruebas y depuración	100	55	5500
Redacción	100	30	3000
Total			38500

Tabla 5. Coste de cada tarea realizada

Teniendo en cuenta las horas totales y el coste total se obtiene un precio por hora de 47,88 €/hora.

1.1.2 HERRAMIENTAS Y SOFTWARE

Aquellos elementos necesarios que son necesarios para el desarrollo del proyecto. Puesto que se trata de un proyecto de desarrollo de aplicación no tiene elementos o componentes que constituyan puramente parte activa del proyecto.

Para realizar un estudio correcto hay que contabilizar no solo el número de unidades de cada recurso y su coste, sino que también es importante tener en cuenta las horas que se utiliza cada uno en el proyecto, las que se utilizan a lo largo del año o la amortización de cada elemento.

Todos estos factores se recogen en la Ecuación (1) que se muestra a continuación:

$$C_t = C_u \cdot n \cdot (t_p / t_a) \cdot a \qquad (1)$$

C_t es el coste total, que es el que interesa finalmente, C_u es el coste unitario del elemento en cuestión, n es el número de unidades, t_p es el tiempo de uso dedicado al proyecto, t_a es el tiempo de uso a lo largo de todo un año y a es la amortización anual.

Quedando de la siguiente manera el presupuesto de esta parte.

Elemento	Ud.	Coste unitario (€/ud)	Horas en proyecto	Horas anuales	Amortización anual (%)	Coste total (€)
Sistema operativo Windows 8.1	1	120	800	1825	25	13,15
Microsoft Office	1	200	130	1000	33	8,58
Ordenador	1	650	800	1825	25	71,23
Visual Studio Professional	1	640	450	100	33	965,25
Total						1058,21

Tabla 6. Coste de "Herramientas y software"

1.1.3 TOTAL DE COSTES DIRECTOS

Por tanto sumando las dos partes que conforman los costes directos de este proyecto se obtiene un total de 39558,21 €, lo cual supone la gran parte de presupuesto total.

1.2 COSTES INDIRECTOS

Este tipo de costes engloba aquellos costes que de esos recursos o elementos que aunque son necesarios para el desarrollo del proyecto, no tienen una relación directa con este.

En el caso de este proyecto se tendrán en cuenta únicamente el consumo eléctrico y el internet necesarios para el uso de los aparatos electrónicos y herramientas del proyecto.

Recurso	Meses	Precio mensual (€/mes)	Precio total (€)
Consumo eléctrico	9	22	198
Internet	9	19	171
Total			269

Tabla 7. Costes indirectos

1.3 COSTES TOTALES

Teniendo en cuenta los cálculos realizados se obtiene que el presupuesto total del proyecto es el siguiente.

Costes directos	Mano de obra	38500 €	39558,21 €
	Herramientas y software	1058,21 €	
Costes indirectos			269 €
Presupuesto total			**39827,2 €**

Tabla 8. Costes totales

Parte III CÓDIGO FUENTE

Capítulo 1 CÓDIGO FUENTE 1

En este capítulo se muestra el código fuente final de todos los archivos utilizados en el proyecto. En el apartado 3.7.1 de "Integración" de la memoria se explica qué realiza cada sección de código y la relación y dependencia que existe entre ellas.

Se dispone de 6 archivos ".cpp" que se muestran con sus respectivos ficheros de cabecera ".h".

Main.h

```
//Main.h

#ifndef MAIN
#define MAIN

#include<opencv2/core/core.hpp>
#include<opencv2/highgui/highgui.hpp>
#include<opencv2/imgproc/imgproc.hpp>
#include<opencv2/ml/ml.hpp>
#include <stdio.h>
#include<iostream>
#include<conio.h>
#include "DetectChars.h"
#include "ExtractData.h"


const cv::Scalar SCALAR_BLACK = cv::Scalar(0.0, 0.0, 0.0);
const cv::Scalar SCALAR_WHITE = cv::Scalar(255.0, 255.0, 255.0);
const cv::Scalar SCALAR_YELLOW = cv::Scalar(0.0, 255.0, 255.0);
const cv::Scalar SCALAR_GREEN = cv::Scalar(0.0, 255.0, 0.0);
const cv::Scalar SCALAR_RED = cv::Scalar(0.0, 0.0, 255.0);



int main(void);


# endif      // MAIN.h
```

Main.cpp

```
//Main.c++

#include "Main.h"

int main(void){
	char reconoce[256] = { 0 };		//variables para guardar lo
	char quiero[256] = { 0 };		//introducido
	int salir = 0;

	char nombreimagen[512] = { 0 };
	char nombremodelo[512] = { 0 };
	int numero = 0;
	size_t iLen = 0;


	bool blnKNNTrainingSuccessful = loadKNNDataAndTrainKNN();
	// intento de cargar el entrenamiento KNN

	if (blnKNNTrainingSuccessful == false) {
	// si no se consigue con éxito
		// mostrar un mensaje de error
		std::cout << std::endl << std::endl << "error: el
entrenamiento KNN no ha tenido exito" << std::endl << std::endl;
		return(0);
// y salir del programa completo
	}

	while (1)
	{

		std::wcout << std::endl << "Paso 1: Introduce el nombre de
su imagen: " << std::endl;
		gets_s(nombreimagen, 512);	//guarda lo introducido en
pszBuffer
		iLen = strlen(nombreimagen);	//tamaño de la cadena
introducida
		nombreimagen[iLen] = '\0';

		while (1){
			std::wcout << std::endl << "Paso 1: Introduce el
modelo de tipo de documento: " << std::endl;
			std::wcout << std::endl << "Para factura pulse 1 "
<< std::endl;
			std::wcout << std::endl << "Para contrato pulse 2 "
<< std::endl;
			gets_s(nombremodelo, 512);	//guarda lo
introducido en pszBuffer
			iLen = strlen(nombremodelo);	//tamaño de la cadena
introducida
```

```
                    numero = atoi(nombremodelo);

                    if (numero == 1){
                          break;
                    }
                    std::cout << std::endl << "Por favor introduce el
numero de un modelo registrado";
             }

             salir = ExtractData(nombreimagen, numero);


             std::cout << std::endl << "Se ha terminado el programa de
reconocimiento " << std::endl;
             _getch();

             if (salir)
                    break;


      }
      return 0;

}
```

ExtractData.h

```
//ExtractData.h

#ifndef EXTRACTDATA
#define EXTRACTDATA

#include <windows.h>
#include <stdio.h>
#include <conio.h>
#include <string.h>
#include<opencv2/core/core.hpp>
#include<opencv2/highgui/highgui.hpp>
#include<opencv2/imgproc/imgproc.hpp>
#include<opencv2/ml/ml.hpp>
#include<iostream>
#include<sstream>
#include <vector>
#include <algorithm>

#include "DetectChars.h"
#include "Correspondencia.h"
```

```
#include "C:\Program Files (x86)\Dynamsoft\Barcode Reader
6.0\Components\C_C++\Include/DynamsoftBarcodeReader.h"

#ifdef _WIN64
#pragma comment(lib, "C:/Program Files (x86)/Dynamsoft/Barcode Reader
6.0/Components/C_C++/Lib/DBRx64.lib")
#else
#pragma comment(lib, "C:\Program Files (x86)\Dynamsoft\Barcode Reader
6.0\Components/C_C++/Lib/DBRx86.lib")
#endif

int ExtractData(char *Imagen, int Modelo);

# endif       // ExtractData.h
```

ExtractData.cpp

```
#include "ExtractData.h"


int ExtractData(char *Imagen, int Modelo)
{

	char reconoce[256] = { 0 };
	char quiero[256] = { 0 };
	char correccion[512] = { 0 };
	int pasos = 0;
	int tamaño = 0;
	char Emisor1[512] = "COMISION NACIONAL DE LOS MERCADOS Y LA
COMPETENCIA";
	char Emisor2[512] = "COMISION NACIONAL DEL MERCADO DE VALORES";
	char Emisor3[512] = "COMISION DEL MERCADO DE LAS
TELECOMUNICACIONES";
	char Productor1[512] = "MARTIFER SOLAR SISTEMAS SOLARES S.A.";
	char Productor2[512] = "ASOCIACION EMPRESARIAL EOLICA";
	char Productor3[512] = "ESDRAS AUTOMATICA S.L.";
	char Decimal[512] = { 0 };


	//Se van a reconocer los datos que mas interesán
	//Podrían reconocerse el resto que datos que hay en la imagen
	std::string datEmisor;                    //nombre entidad emisora
	std::string datProductor;         //nombre entidad productora
	std::string NIF_Em;                       //NIF entidad emisora
	std::string NIF_Pr;                       //NIF entidad emisora
	std::string Precio;                       //Precio total
	std::string Numero;                       //numero del documento

	cv::Mat imgOriginalScene;           // imagen original
```

```
	imgOriginalScene = cv::imread(Imagen);		// abrir la
imagen deseada

	if (imgOriginalScene.empty()) {				// y
si no se puede abrir
		std::cout << "error: imagen no leida \n\n";	//
mostrar un mensaje de error
		_getch();
// esperar a pulsar una tecla para salir del programa
		return 0;					//
salir
	}

	cv::Mat imgROIEmisor;
	cv::Mat imgROIProductor;
	cv::Mat imgROINIF_Em;
	cv::Mat imgROINIF_Pr;
	cv::Mat imgROIPrecio;
	cv::Mat imgROINumero;

	switch (Modelo){
	//Se hacen los recortes de la region de interes
	case 1:
		//En caso de tener otros tipos de documentos las regiones
de interes serían otras
		cv::Rect Emisor(440, 545, 670, 75);
		imgROIEmisor = imgOriginalScene(Emisor);
		cv::Rect Productor(1430, 550, 450, 47);
		imgROIProductor = imgOriginalScene(Productor);
		cv::Rect NIF_em(440, 850, 400, 60);
		imgROINIF_Em = imgOriginalScene(NIF_em);
		cv::Rect NIF_pr(2050, 245, 300, 50);
		imgROINIF_Pr = imgOriginalScene(NIF_pr);
		cv::Rect precio(2235, 1450, 120, 50);
		imgROIPrecio = imgOriginalScene(precio);
		cv::Rect numero(1700, 290, 350, 45);
		imgROINumero = imgOriginalScene(numero);
	}

	std::cout << std::endl << "De que quiere realizar el
reconocimiento? " << std::endl;
	std::cout << std::endl << "Para todo pulse ---------------- t"
<< std::endl;
	std::cout << std::endl << "Para entidad emisora pulse ----- e"
<< std::endl;
	std::cout << std::endl << "Para entidad productora pulse -- p"
<< std::endl;
	std::cout << std::endl << "Para NIF de la entidad emisora pulse
-- d" << std::endl;
	std::cout << std::endl << "Para NIF de la entidad productora
pulse -- n" << std::endl;
	std::cout << std::endl << "Para el precio pulse -- m" <<
std::endl;
	std::cout << std::endl << "Para el numero de documento pulse --
x" << std::endl;
```

```
	gets_s(reconoce, 256);	//guarda lo introducido en reconoce

	//se reconoce los caracteres de la región de interes y se
muestran
	if (strlen(reconoce) == 1 && (reconoce[0] == 'e' || reconoce[0]
== 'E' || reconoce[0] == 't'))
	{
		datEmisor = recognizeCharsInPlate(imgROIEmisor);
		std::cout << std::endl << "El nombre de la entidad emisora
es = " << datEmisor << std::endl;
	}

	if (strlen(reconoce) == 1 && (reconoce[0] == 'p' || reconoce[0]
== 'P' || reconoce[0] == 't'))
	{
		std::string datProductor =
recognizeCharsInPlate(imgROIProductor);
		std::cout << std::endl << "El nombre de la entidad
productora es = " << datProductor << std::endl;
	}

	if (strlen(reconoce) == 1 && (reconoce[0] == 'd' || reconoce[0]
== 'D' || reconoce[0] == 't'))
	{
		NIF_Em = recognizeCharsInPlate(imgROINIF_Em);
		std::cout << std::endl << "El NIF de la entidad emisora es
= " << NIF_Em << std::endl;
	}

	if (strlen(reconoce) == 1 && (reconoce[0] == 'n' || reconoce[0]
== 'N' || reconoce[0] == 't'))
	{
		NIF_Pr = recognizeCharsInPlate(imgROINIF_Pr);
		std::cout << std::endl << "El NIF de la entidad productora
es = " << NIF_Pr << std::endl;
	}

	if (strlen(reconoce) == 1 && (reconoce[0] == 'm' || reconoce[0]
== 'M' || reconoce[0] == 't'))
	{
		Precio = recognizeCharsInPlate(imgROIPrecio);
		strcpy_s(Decimal, Precio.c_str());
		tamaño = strlen(Decimal);
		// esto es necesario porque la coma "," no la reconoce
bien, y, puesto que se sabe que siempre va a venir el numero con dos
decimales, se le pone manualmente
		Decimal[tamaño + 1] = Decimal[tamaño];
		Decimal[tamaño] = Decimal[tamaño - 1];
		Decimal[tamaño - 1] = Decimal[tamaño - 2];
		Decimal[tamaño-2] = ',';
		std::cout << std::endl << "El precio total es = " <<
Decimal << std::endl;
	}
```

```
	if (strlen(reconoce) == 1 && (reconoce[0] == 'x' || reconoce[0] 
== 'X' || reconoce[0] == 't'))
	{
		Numero = recognizeCharsInPlate(imgROINumero);
		std::cout << std::endl << "El numero del documento = " << 
Numero << std::endl;
	}

	_getch();		//pausa

////////////////////////////////////
	std::cout << std::endl << "Quiere comprobar si el reconocimiento 
ha sido correcto con toda la informacion de los codigos de barras" << 
std::endl;
	std::cout << std::endl << "s=si / n=no" << std::endl;
	gets_s(quiero, 256);	//guarda lo introducido en quiero
	//se busca la información correspondiente en los códigos de 
barras
	if (strlen(quiero) == 1 && (quiero[0] == 's' || quiero[0] == 
's')){
		std::cout << std::endl << "Quieres ver paso a paso el 
reconocimiento de cada codigo?" << std::endl;
		std::cout << std::endl << "s=si / n=no" << std::endl;
		gets_s(quiero, 256);

		if (strlen(quiero) == 1 && (quiero[0] == 's' || quiero[0] 
== 's'))
			pasos = 1;
		if (strlen(reconoce) == 1 && (reconoce[0] == 'e' || 
reconoce[0] == 'E' || reconoce[0] == 't')){
			strcpy_s(correccion, (comprueba('e', datEmisor, 
pasos)).c_str());
			std::cout << std::endl << "Entidad emisora 
correcta: " << correccion << std::endl;
		}

		if (strlen(reconoce) == 1 && (reconoce[0] == 'p' || 
reconoce[0] == 'P' || reconoce[0] == 't'))
		{
			strcpy_s(correccion, (comprueba('p', datProductor, 
pasos)).c_str());
			std::cout << std::endl << "Entidad productora 
correcta: " << correccion << std::endl;
		}

		if (strlen(reconoce) == 1 && (reconoce[0] == 'd' || 
reconoce[0] == 'D' || reconoce[0] == 't')){
			strcpy_s(correccion, (comprueba('d', NIF_Em, 
pasos)).c_str());
			std::cout << std::endl << "NIF de la entidad de la 
emisora: " << correccion << std::endl;
		}
```

```
            if (strlen(reconoce) == 1 && (reconoce[0] == 'n' ||
reconoce[0] == 'N' || reconoce[0] == 't'))
            {
                  strcpy_s(correccion, (comprueba('n', NIF_Pr,
pasos)).c_str());
                  std::cout << std::endl << "NIF de la entidad de la
productora: " << correccion << std::endl;
            }

            if (strlen(reconoce) == 1 && (reconoce[0] == 'm' ||
reconoce[0] == 'M' || reconoce[0] == 't'))
            {
                  strcpy_s(correccion, (comprueba('m', Precio,
pasos)).c_str());
                  std::cout << std::endl << "El precio total en los
códigos de barras es: " << correccion << std::endl;
            }

            if (strlen(reconoce) == 1 && (reconoce[0] == 'x' ||
reconoce[0] == 'X' || reconoce[0] == 't'))
            {
                  strcpy_s(correccion, (comprueba('x', Numero,
pasos)).c_str());
                  std::cout << std::endl << "El numero reconocido en
los códigos de barras es: " << correccion << std::endl;
            }
      }

      std::cout << std::endl << "Quiere comprobar si el reconocimiento
ha sido correcto con la informacion de los posibles nombres" <<
std::endl;
      std::cout << std::endl << "s=si / n=no" << std::endl;
      gets_s(quiero, 256);
      //se busca la relación con la información base guardada
      if (strlen(quiero) == 1 && (quiero[0] == 's' || quiero[0] ==
's')){
            if (strlen(reconoce) == 1 && (reconoce[0] == 'd' ||
reconoce[0] == 'D' || reconoce[0] == 'n' || reconoce[0] == 'N' ||
reconoce[0] == 'm' || reconoce[0] == 'M' || reconoce[0] == 'x' ||
reconoce[0] == 'X')){
                  std::cout << std::endl << "No hay informacion base
sobre este dato" << std::endl;
            }//de este tipo de datos al poder ser cualquier cosa no se
dispone de datos posibles

            if (strlen(reconoce) == 1 && (reconoce[0] == 'e' ||
reconoce[0] == 'E' || reconoce[0] == 't')){
                  if (distanciaentrecadenas(datEmisor.c_str(),
Emisor1) < distanciaentrecadenas(datEmisor.c_str(), Emisor2) &&
                        distanciaentrecadenas(datEmisor.c_str(),
Emisor1) < distanciaentrecadenas(datEmisor.c_str(), Emisor3)){
                        std::cout << std::endl << "A la cadena que
mas se parece es a  " << Emisor1 << std::endl;
                  }
```

```
                    else if (distanciaentrecadenas(datEmisor.c_str(),
Emisor2) < distanciaentrecadenas(datEmisor.c_str(), Emisor2)){
                            std::cout << std::endl << "A la cadena que
mas se parece es a  " << Emisor2 << std::endl;
                    }
                    else{
                            std::cout << std::endl << "A la cadena que
mas se parece es a  " << Emisor3 << std::endl;
                    }
            }

            if (strlen(reconoce) == 1 && (reconoce[0] == 'p' ||
reconoce[0] == 'P' || reconoce[0] == 't')){
                    if (distanciaentrecadenas(datProductor.c_str(),
Productor1) < distanciaentrecadenas(datProductor.c_str(), Productor2)
&&
                            distanciaentrecadenas(datProductor.c_str(),
Productor1) < distanciaentrecadenas(datProductor.c_str(), Productor3)){
                            std::cout << std::endl << "A la cadena que
mas se parece es a  " << Productor1 << std::endl;
                    }
                    else if
(distanciaentrecadenas(datProductor.c_str(), Productor2) <
distanciaentrecadenas(datProductor.c_str(), Productor2)){
                            std::cout << std::endl << "A la cadena que
mas se parece es a  " << Productor2 << std::endl;
                    }
                    else{
                            std::cout << std::endl << "A la cadena que
mas se parece es a  " << Productor3 << std::endl;
                    }
            }
            _getch();
      }
      return 1;
}
```

DetectChars.h

```
// DetectChars.h
#ifndef DETECT_CHARS_H
#define DETECT_CHARS_H
#include "PossibleChar.h"
#include "Preprocesamiento.h"

const int RESIZED_CHAR_IMAGE_WIDTH = 20;
const int RESIZED_CHAR_IMAGE_HEIGHT = 30;
extern cv::Ptr<cv::ml::KNearest>  kNearest;

bool loadKNNDataAndTrainKNN(void);

std::string recognizeCharsInPlate(cv::Mat &imgOriginal);
#endif // DETECT_CHARS
```

DetectChars.cpp

```
// DetectChars.cpp

#include "DetectChars.h"

//variables globales
///////////////////////////////////////////////////////////////////
cv::Ptr<cv::ml::KNearest> kNearest = cv::ml::KNearest::create();

///////////////////////////////////////////////////////////////////
bool loadKNNDataAndTrainKNN(void) {

	// leer el entrenamiento de clasificaciones
/////////////////////////////////////////////////////
	cv::Mat matClassificationInts;		// vector en el que
leeremos los numeros de clasificaciones

	cv::FileStorage fsClassifications("classifications.xml",
cv::FileStorage::READ);		// abrir el archivo de clasificaciones

	if (fsClassifications.isOpened() == false) {
// si no se abre
		std::cout << "Error,incapaz de abir el archivo de
clasificaciones, saliendo del programa\n\n";		// mostrar mensaje
de error
		system("pause");
		cv::waitKey();
		return(false);
// salir
	}

	fsClassifications["classifications"] >> matClassificationInts;
// leer la sección de clasificaciones en la matriz matClassificationInts
	fsClassifications.release();
// cerrar el archivo de clasificaciones

	// leer el entrenamiento de imagenes
/////////////////////////////////////////////////////////

	cv::Mat matTrainingImagesAsFlattenedFloats;		// leeremos
muchas imagenes en esta variable solamente como si fuera un vector

	cv::FileStorage fsTrainingImages("images.xml",
cv::FileStorage::READ);		//abrir el archivo de entrenamiento
de imagenes

	if (fsTrainingImages.isOpened() == false) {
//si no se abre
		std::cout << "Error,incapaz de abir el archivo de
imagenes, saliendo del programa\n\n";		//mostrar mensaje de error
		return(false);
// salir
	}
```

```
	fsTrainingImages["images"] >> matTrainingImagesAsFlattenedFloats;
//  leer la sección de clasificaciones en la matriz de entrenamiento de
imagenes
	fsTrainingImages.release();
// cerrar el archivo de entrenamiento de imagenes

	// entrenamiento
////////////////////////////////////////////////////////////////////////
///////

	// llamamos al entrenamiento. Los parametros son una sola matriz
aunque en realidad son muchos imagenes/numeros
	kNearest->setDefaultK(1);

	kNearest->train(matTrainingImagesAsFlattenedFloats,
cv::ml::ROW_SAMPLE, matClassificationInts);

	return true;
}

std::string recognizeCharsInPlate(cv::Mat &imgOriginal) {

	cv::Mat imgGrayscale;
	cv::Mat imgThresh;
	preprocesamiento(imgOriginal, imgGrayscale, imgThresh);

	std::vector<PossibleChar> vectorOfMatchingChars;
//vector que tendrá los posibles caracteres
	std::vector<std::vector<cv::Point> > contours;
	//cv::imshow("imgGrayscale", imgGrayscale);
	//cv::imshow("imgThresh", imgThresh);
	//cv::waitKey(100);    // necesario para que se vea el imgThresh
de cada uno
	////// si no lo pones se ve solo el del primero

	cv::Mat imgThreshCopy;

	imgThreshCopy = imgThresh.clone();
// hacer una copia de imgThresh necesaria para findcountours


	std::vector<cv::Vec4i> v4iHierarchy;
// declaramos un vector para la jerarquía
	cv::findContours(imgThreshCopy,             // img entrada, estar
seguros de que es una copia
		contours,                                // contornos de
salida
		v4iHierarchy,                              // jerarquía de
salida
		cv::RETR_EXTERNAL,                         // recupera los
contornos exteriores solo
		cv::CHAIN_APPROX_SIMPLE);                    // compresión
horizontal, vertical, y diagonal de los segmentos y dejamos solo los
puntos finales
```

```
	for (auto &contour : contours) {                                //
para cada contorno
		PossibleChar possibleChar(contour);

		if (checkIfPossibleChar(possibleChar)) {                //
si el contorno es un caracter posible
			vectorOfMatchingChars.push_back(possibleChar);
// añadir al vector de posibles caracteres
		}
	}
	std::string strChars;               // cadena final que
devolveremos

	std::string strChars1;
	std::string strChars2;

	cv::Mat imgThreshColor;

	// ordenar caracteres de izquierda a derecha
	std::sort(vectorOfMatchingChars.begin(),
vectorOfMatchingChars.end(), PossibleChar::sortCharsLeftToRight);

	cv::cvtColor(imgThresh, imgThreshColor, CV_GRAY2BGR);       //
volver a poner color en imgThresh y asi dibujar los contornos en color
en ella
	//cv::imshow("imgThreshColor", imgThreshColor);
	//cv::waitKey(100);    // necesario para que se vea el imgThresh
de cada uno
	//// si no lo pones se ve solo el del primero

	for (int i = 0; i < vectorOfMatchingChars.size(); i++) {
// para cada contorno

		// dibujar un contorno verde alrededor del caracter
		cv::rectangle(imgOriginal,
// dibujar rectangulo en la imagen original
			vectorOfMatchingChars[i].boundingRect,
// rectangulo a dibujar
			cv::Scalar(0, 255, 0),
// verde
			2);
// grosor

		cv::Mat matROI =
imgThresh(vectorOfMatchingChars[i].boundingRect);      // obtener la
imagen ROI del rectangulo

		cv::Mat matROIResized;
		cv::resize(matROI,                          matROIResized,
cv::Size(RESIZED_CHAR_IMAGE_WIDTH,          RESIZED_CHAR_IMAGE_HEIGHT));
//  redimensionar  la  imgen,  esto  será  más  consistente  para  el
reconocimiento y almacenamiento
```

```
                cv::Mat matROIFloat;
                matROIResized.convertTo(matROIFloat, CV_32FC1);
// convertir de Mat a float, necesario para llamar a find_nearest

                cv::Mat matROIFlattenedFloat = matROIFloat.reshape(1, 1);

                cv::Mat matCurrentChar(0, 0, CV_32F);

                kNearest->findNearest(matROIFlattenedFloat, 1,
matCurrentChar);     // ya podemos llamar a find_nearest !!!

                float fltCurrentChar = (float)matCurrentChar.at<float>(0,
0);
                // añadimos el caracter a la cadena completa



                if (vectorOfMatchingChars[i].intCenterY <
(imgOriginal.rows * 0.5)) {
                        strChars1 = strChars1 + char(int(fltCurrentChar));
                }
                else{
                        strChars2 = strChars2 + char(int(fltCurrentChar));
                }


        }
        //cv::imshow("imgOriginal", imgOriginal);

        //cv::waitKey(100);

        strChars = strChars1 + strChars2;

        return(strChars);               // devolver resultado
}
```

Pre-procesamiento

```
// Preprocesamiento.h

#ifndef PREPROCESAMIENTO_H
#define PREPROCESAMIENTO_H

#include<opencv2/core/core.hpp>
#include<opencv2/highgui/highgui.hpp>
#include<opencv2/imgproc/imgproc.hpp>

///////variables globales
const cv::Size GAUSSIAN_SMOOTH_FILTER_SIZE = cv::Size(5, 5);
const int ADAPTIVE_THRESH_BLOCK_SIZE = 11;
const int ADAPTIVE_THRESH_WEIGHT = 2;
```

```
// funciones
//////////////////////////////////////////////////////////////////

void preprocesamiento(cv::Mat &imgOriginal, cv::Mat &imgGrayscale,
cv::Mat &imgThresh);

cv::Mat extractValue(cv::Mat &imgOriginal);

cv::Mat maximizeContrast(cv::Mat &imgGrayscale);

#endif // PREPROCESAMIENTO_H
```

Pre-procesamiento.cpp

```
// Preprocesamiento.cpp

#include "Preprocesamiento.h"

///////////////////////////////////////////////////////////////////////////
////////////////////////////
void preprocesamiento(cv::Mat &imgOriginal, cv::Mat &imgGrayscale,
cv::Mat &imgThresh) {

	//cv::imshow("emisororiginal", imgOriginal);

	//cv::waitKey(100);
	// lo que viene a continuacion de imwrite se usa para guarder las
imagenes y poder usarlas en la memoria
	//cv::imwrite("C:/Users/user/ICAI/Cuarto/Tfg/figuras
memoria/emisororiginal.jpg", imgOriginal);

	imgGrayscale = extractValue(imgOriginal);
// llamar a la función para obtener la imgagen en escala de grises

	cv::Mat imgMaxContrastGrayscale = maximizeContrast(imgGrayscale);
// maximizar el contraste

	cv::Mat imgBlurred;
	cv::GaussianBlur(imgMaxContrastGrayscale, imgBlurred,
GAUSSIAN_SMOOTH_FILTER_SIZE, 0);		// emborronamiento gausiano

	cv::adaptiveThreshold(imgBlurred, imgThresh, 255.0,
CV_ADAPTIVE_THRESH_GAUSSIAN_C, CV_THRESH_BINARY_INV,
ADAPTIVE_THRESH_BLOCK_SIZE, ADAPTIVE_THRESH_WEIGHT);
	//cv::imshow("emisorthresh", imgThresh);

	//cv::waitKey(100);
	//cv::imwrite("C:/Users/user/ICAI/Cuarto/Tfg/figuras
memoria/emisorthresh.jpg", imgThresh);
}
```

```
///////////////////////////////////////////////////////////////////////
cv::Mat extractValue(cv::Mat &imgOriginal) {
	cv::Mat imgHSV;
	std::vector<cv::Mat> vectorOfHSVImages;
	cv::Mat imgValue;

	cv::cvtColor(imgOriginal, imgHSV, CV_BGR2HSV);

	cv::split(imgHSV, vectorOfHSVImages);

	imgValue = vectorOfHSVImages[2];

	return(imgValue);
}

///////////////////////////////////////////////////////////////////////
cv::Mat maximizeContrast(cv::Mat &imgGrayscale) {
	cv::Mat imgTopHat;
	cv::Mat imgBlackHat;
	cv::Mat imgGrayscalePlusTopHat;
	cv::Mat imgGrayscalePlusTopHatMinusBlackHat;

	cv::Mat structuringElement =
cv::getStructuringElement(CV_SHAPE_RECT, cv::Size(3, 3));
	cv::morphologyEx(imgGrayscale, imgTopHat, CV_MOP_TOPHAT,
structuringElement);
	cv::morphologyEx(imgGrayscale, imgBlackHat, CV_MOP_BLACKHAT,
structuringElement);
	imgGrayscalePlusTopHat = imgGrayscale + imgTopHat;
	imgGrayscalePlusTopHatMinusBlackHat = imgGrayscalePlusTopHat -
imgBlackHat;

	//cv::imshow("imgcontrast", imgGrayscalePlusTopHatMinusBlackHat);
	//cv::waitKey(100);
	//cv::imwrite("C:/Users/user/ICAI/Cuarto/Tfg/figuras
memoria/emisorcontraste.jpg", imgGrayscalePlusTopHatMinusBlackHat);

	return(imgGrayscalePlusTopHatMinusBlackHat);
}
```

PossibleChar.h

```
//Possible char.h

#ifndef POSSIBLE_CHAR
#define POSSIBLE_CHAR

#include<opencv2/core/core.hpp>
#include<opencv2/highgui/highgui.hpp>
#include<opencv2/imgproc/imgproc.hpp>
#include<opencv2/ml/ml.hpp>
```

```
#include<iostream>
#include<conio.h>

// constantes para checkIfPossibleChar, esto prueba a encontrar un
caracter similar (no compara)
const int MIN_PIXEL_WIDTH = 2;
const int MIN_PIXEL_HEIGHT = 12;
const double MIN_ASPECT_RATIO = 0.25;
const double MAX_ASPECT_RATIO = 1.0;
const int MIN_PIXEL_AREA = 80;

class PossibleChar {
public:
	// variables miembro
///////////////////////////////////////////////////////////////////////////////
	std::vector<cv::Point> contour;

	cv::Rect boundingRect;

	int intCenterX;
	int intCenterY;

	double dblDiagonalSize;
	double dblAspectRatio;

	/////////////////////////////////////////////////////////////////
	static bool sortCharsLeftToRight(const PossibleChar &pcLeft,
const PossibleChar & pcRight) {
		return(pcLeft.intCenterX < pcRight.intCenterX);
	}

	/////////////////////////////////////////////////////////////////
	bool operator == (const PossibleChar& otherPossibleChar) const {
		if (this->contour == otherPossibleChar.contour) return
true;
		else return false;
	}

	/////////////////////////////////////////////////////////////////
	bool operator != (const PossibleChar& otherPossibleChar) const {
		if (this->contour != otherPossibleChar.contour) return
true;
		else return false;
	}

	// prototipos de funciones
///////////////////////////////////////////////////////////////////////////////
	PossibleChar(std::vector<cv::Point> _contour);

};

bool checkIfPossibleChar(PossibleChar &possibleChar);

#endif  // POSSIBLE_CHAR
```

DetectChars.cpp

```
// PossibleChar.cpp

#include "PossibleChar.h"

///////////////////////////////////////////////////////////////////////////////
PossibleChar::PossibleChar(std::vector<cv::Point> _contour) {
	contour = _contour;

	boundingRect = cv::boundingRect(contour);

	intCenterX = (boundingRect.x + boundingRect.x + boundingRect.width) / 2;
	intCenterY = (boundingRect.y + boundingRect.y + boundingRect.height) / 2;

	dblDiagonalSize = sqrt(pow(boundingRect.width, 2) + pow(boundingRect.height, 2));

	dblAspectRatio = (float)boundingRect.width / (float)boundingRect.height;
}

bool checkIfPossibleChar(PossibleChar &possibleChar) {
	// esta función es solo una pequeña comprobación que mira las dimensiones del contorno para ver si puede ser un caracter
	// no hace comparación entre caracteres

	if (possibleChar.boundingRect.area() > MIN_PIXEL_AREA &&
		possibleChar.boundingRect.width > MIN_PIXEL_WIDTH &&
		possibleChar.boundingRect.height > MIN_PIXEL_HEIGHT &&
		MIN_ASPECT_RATIO < possibleChar.dblAspectRatio &&
		possibleChar.dblAspectRatio < MAX_ASPECT_RATIO) {
		return(true);
	}
	else {
		return(false);
	}
}
```

Correspondencia.h

```
//Correspondencia.h

#ifndef CORRESPONDENCIA
#define CORRESPONDENCIA

#include <windows.h>
#include <stdio.h>
#include <conio.h>
#include <string.h>
#include <locale.h>
#include <wchar.h>

#include <vector>
#include <algorithm>

#include<iostream>
#include<sstream>


#include "C:\Program Files (x86)\Dynamsoft\Barcode Reader
6.0\Components\C_C++\Include/DynamsoftBarcodeReader.h"

#ifdef _WIN64
#pragma comment(lib, "C:/Program Files (x86)/Dynamsoft/Barcode Reader
6.0/Components/C_C++/Lib/DBRx64.lib")
#else
#pragma comment(lib, "C:\Program Files (x86)\Dynamsoft\Barcode Reader
6.0\Components/C_C++/Lib/DBRx86.lib")
#endif

int distanciaentrecadenas(const char *s1, char *s2);
const char* ObtenerQuiero(char quiero);
void ToHexString(unsigned char* pSrc, int iLen, char* pDest);
std::string comprueba(char letra, std::string parecido, int pasos);

# endif       // Correspondencia.h
```

Correspondencia.cpp

```
//Correspondencia.cpp
#include "Correspondencia.h"

int distanciaentrecadenas(const char *s1, char *s2){
	using namespace std;


	int N1 = strlen(s1);
	int N2 = strlen(s2);
	int i, j;
	vector<int> T(N2 + 1);

	for (i = 0; i <= N2; i++)
		T[i] = i;

	for (i = 0; i < N1; i++)
	{
		T[0] = i + 1;
		int corner = i;
		for (j = 0; j < N2; j++)
		{
			int upper = T[j + 1];
			if (s1[i] == s2[j])
				T[j + 1] = corner;
			else
				T[j + 1] = min(T[j], min(upper, corner)) + 1;
			corner = upper;
		}
	}
	return T[N2];
}

const char* ObtenerQuiero(char quiero)
{
	switch (quiero)
	{
	case 'e':
	case 'd':
		return "SellerParty";
	case 'p':
	case 'n':
		return "BuyerParty";
	case 'm':
	case 'x':
		return "FileHeader";
	default:
		return NULL;
	}
}
```

```
std::string comprueba(char letra, std::string parecido, int pasos)
{
	setlocale(LC_ALL, "");

	const char* pszSettingFile = "C:\\Program Files
(x86)\\Dynamsoft\\Barcode Reader 6.0\\Templates\\default_settings.json";
	const char* pszTemplateName = NULL;

	char pszBuffer[512] = { 0 };
	char pszImageFile[512] = { 0 };

	//variables para el reconocimiento de los códigos de barras
	char str[512] = "C:\\Users\\User\\Desktop\\IMAGENES PDF
TFG\\x.jpg";
	char str1[512] = "C:\\Users\\User\\Desktop\\IMAGENES PDF
TFG\\xx.JPG";
	int m = 32768;
	int n = 32768;
	int i = 1;
	char Cadena[4096] = { 0 };
	char Final[32768] = { 0 };
	char inicio[512] = { 0 };
	size_t Long = 0;
	int vend = 0;
	int comp = 0;
	float segundos = 0;

	//variables para encontrar la información de interés
	char copia[1024] = { 0 };
	char nombre[1024] = { 0 };
	char numero[1024] = "BatchIdentifier";
	char entidad[1024] = "CorporateName";
	char nif[1024] = "TaxIdentificationNumber";
	char precio[1024] = "TotalAmount";
	char cadena[4096] = { 0 };
	int l = 0;
	int j = 0;
	int k = 0;
	int correcto = 0;
	int fin = 0;


	int iIndex = 0;
	int iRet = -1;
	char * pszTemp = NULL;
	char * pszTemp1 = NULL;
	char * pszTemp2 = NULL;
	unsigned __int64 ullTimeBegin = 0;
	unsigned __int64 ullTimeEnd = 0;
	size_t iLen = 0;
	FILE* fp = NULL;
	int iExitFlag = 0;
	char szErrorMsg[256];
```

```
strcpy_s(copia, ObtenerQuiero(letra));

	//lo que viene a continuacion también se puede hacer con una
funcion como la de ObtenerQuiero
	//pero así lo tenemos de dos maneras diferentes
	if (letra == 'e' || letra == 'p' )
	{
		strcpy_s(nombre, entidad);
	}
	if (letra == 'd' || letra == 'n' )
	{
		strcpy_s(nombre, nif);
	}
	if (letra == 'm')
	{
		strcpy_s(nombre, precio);
	}
	if (letra == 'x')
	{
		strcpy_s(nombre, numero);
	}

	CBarcodeReader reader;

	reader.InitLicense("t0068MgAAAEKRJW/TAwoOhPqgdhWJ1k0dsRKPVALwWnHBn
dwHPv1tVUAJWzblySqmMQytsZToZNovsDs40CLXdsKfnmPKG0A=");
	iRet = reader.LoadSettingsFromFile(pszSettingFile, szErrorMsg,
256);
	if (iRet != DBR_OK)
	{
		std::cout << std::endl << "Codigo erroneo: " << iRet <<":
Mensaje de error: " << szErrorMsg << std::endl;
		return "Problema en el reconocimiento";  //
	return -1 originalmente
	}

	while (1)
	{
		while (i<22)
		{
			if (i <= 9)
			{
				char str[512] =
"C:\\Users\\User\\Desktop\\IMAGENES PDF TFG\\x.JPG";
				str[strlen(str)] = '\0';
				str[strlen(str) - 5] = i + '0';
				strcpy_s(pszBuffer, str);
				pszBuffer[strlen(str)] = '\0';

				if (pasos){
					std::cout << std::endl << "La direccion
de su imagen es: " << pszBuffer << std::endl;
				}

				memset(pszImageFile, 0, 512);
```

```
                        if (pszBuffer[0] == '\"' &&
pszBuffer[strlen(str) - 1] == '\"')  //Por si empieza o acaba en "\"
                                memcpy(pszImageFile, &pszBuffer[1],
strlen(str) - 2);
                        else
                                memcpy(pszImageFile, pszBuffer,
strlen(str));
                        errno_t err = 0;
                        err = fopen_s(&fp, pszImageFile, "rb");
                        //si no hay error al abrir la direccion de la
imagen seleccionada
                        if (err == 0)
                        {
                                fclose(fp);   //cerrar el archivo
                                break;
                        }
                }

                if (i >= 9)
                {
                        char str1[512] =
"C:\\Users\\User\\Desktop\\IMAGENES PDF TFG\\xx.JPG";
                        //pide la ubicacion de el archivo de la
imagen
                        str1[strlen(str1)] = '\0';
                        str1[strlen(str1) - 5] = i % 10 + '0';
                        str1[strlen(str1) - 6] = i / 10 + '0';

                        strcpy_s(pszBuffer, str1);
                        pszBuffer[strlen(str1)] = '\0';
                        if (pasos){
                                std::cout << std::endl << "La
direccion de su imagen es: " << pszBuffer << std::endl;
                        }

memset(pszImageFile, 0, 512);
                        if (pszBuffer[0] == '\"' &&
pszBuffer[strlen(str1) - 1] == '\"')  //Por si empieza o acaba en "\"
                                memcpy(pszImageFile, &pszBuffer[1],
strlen(str1) - 2);
                        else
                                memcpy(pszImageFile, pszBuffer,
strlen(str1));
                        errno_t err = 0;
                        err = fopen_s(&fp, pszImageFile, "rb");

                        if (err == 0)
                        {
                                fclose(fp);   //cerrar el archivo
                                break;
                        }
                }

                std::cout << std::endl << "Introduce la
direccion/ruta correcta" << std::endl;
                break;
        }
```

```
		if (i >= 22)
		{
	//lo que viene a continuación tiene sentido cuando solo es un
programa de reconocimiento de códigos, pero dentro del programa final no
tiene sentido ponerlo

			//std::cout << std::endl << " La informacion leida
en la comprobacion es: " << Final << std::endl;
			//std::cout << std::endl << " El tiempo total del
reconocimiento ha sido: " << segundos << std::endl;
			//printf("\r\n>> pulsa q para acabar el
reconocimiento \r\n");
			//gets_s(pszBuffer, 512);
			//iLen = strlen(pszBuffer);
			//pszBuffer[iLen] = '\0';
		}

		if (iLen > 0)
		{
			if (strlen(pszBuffer) == 1 && (pszBuffer[0] == 'q'
|| pszBuffer[0] == 'Q')) // salir si se pulsa q
			{
				iExitFlag = 1;
				break;
			}
		}

		if (iExitFlag)
			break;
		while (1)
		{
			if (i == 22)
				iExitFlag = 1;
			pszTemplateName = "PDF417_DEFAULT";    // tipo de
clase de código de barras que se va a utilizar
			if (pszTemplateName != NULL)
				break;

			std::cout << std::endl << "Por favor elige un
numero valido"  << std::endl;
		}
		if (iExitFlag)
			break;
		if (pasos){
			std::cout << std::endl << "Resultados del codigo de
barras" << std::endl << "-----------------" <<std::endl;
		}

		// Leer el codigo de barras
		ullTimeBegin = GetTickCount();
		iRet = reader.DecodeFile(pszImageFile, pszTemplateName);
		ullTimeEnd = GetTickCount();
		// resulatado de salida del codigo de barras
		pszTemp = (char*)malloc(m); //4096
		pszTemp2 = (char*)malloc(n);
```

```
		if (iRet != DBR_OK && iRet != DBRERR_LICENSE_EXPIRED &&
iRet != DBRERR_QR_LICENSE_INVALID &&
			iRet != DBRERR_1D_LICENSE_INVALID && iRet !=
DBRERR_PDF417_LICENSE_INVALID && iRet !=
DBRERR_DATAMATRIX_LICENSE_INVALID)
		{
			std::cout << std::endl << "Error al leer el código:
" << CBarcodeReader::GetErrorString(iRet) << std::endl;
			continue;
		}
		STextResultArray *paryResult = NULL;
		reader.GetAllTextResults(&paryResult);
		if (paryResult->nResultsCount == 0)
		{
			std::cout << std::endl << "No se ha encontrado el
código. Tiempo total: " << ((float)(ullTimeEnd - ullTimeBegin) / 1000)
<< " segundos." << std::endl;
			CBarcodeReader::FreeTextResults(&paryResult);
			continue;
		}
		segundos = ((float)(ullTimeEnd - ullTimeBegin) / 1000) +
segundos;

		if (pasos){
			std::cout << std::endl << "No se ha encontrado el
código. Tiempo total: " << ((float)(ullTimeEnd - ullTimeBegin) / 1000)
<< " segundos." << std::endl;
			free(pszTemp2);
		}
		for (iIndex = 0; iIndex < paryResult->nResultsCount;
iIndex++)
		{
			if (pasos)
			{

				std::cout << std::endl << "Codigo de barras "
<< i << std::endl;
				std::cout << std::endl << "    Tipo " <<
paryResult->ppResults[iIndex]->pszBarcodeFormatString << std::endl;
				std::cout << std::endl << "Valor: " <<
paryResult->ppResults[iIndex]->pszBarcodeText << std::endl;
			}

			strcpy_s(Cadena, paryResult->ppResults[iIndex]-
>pszBarcodeText);
			Cadena[strlen(Cadena)] = '\0';
			Final[strlen(Final)] = '\0';
			strcat_s(Final, Cadena);
			Final[strlen(Final)] = '\0';
			pszTemp1 = (char*)malloc(paryResult-
>ppResults[iIndex]->nBarcodeBytesLength * 3 + 1);
			ToHexString(paryResult->ppResults[iIndex]-

>pBarcodeBytes,	 paryResult->ppResults[iIndex]->nBarcodeBytesLength,

pszTemp1);
```

```
                        if (pasos){
                                sprintf_s(pszTemp, m, "    Hex Data: %s\r\n",
pszTemp1);
                                printf(pszTemp);
                        }
                        free(pszTemp1);
                }
                free(pszTemp);
                CBarcodeReader::FreeTextResults(&paryResult);
                i++;
        }
        for (l = 0; l < strlen(Final); l++)
                if (Final[l] == copia[0])
                {
                        if (strlen(Final) - l < strlen(copia) || fin == 1)
                                return "error";

                        for (j = 0; j < strlen(copia); j++)
                                if (Final[l + j] != copia[j] || fin == 1)
                                        break;

                        if (j == strlen(copia)){
                                correcto = 1;
                                if (correcto)
                                {
                                        for (l = (l + j); l < strlen(Final);
l++)
                                                if (Final[l] == nombre[0])
                                                {
                                                        if (strlen(Final) - l <
strlen(nombre))
                                                                return "error";
                                                        for (j = 0; j <
strlen(nombre); j++)
                                                                if (Final[l + j]
!= nombre[j] || fin == 1)
                                                                        break;

                                                        if (j ==
strlen(nombre)){
                                                                correcto = 1;
                                                                //std::cout <<
std::endl << " el nombre leido de " << nombre << std::endl;
                                                                //_getch();

                                                                if (correcto)
                                                                {
                                                                        while
(Final[l + j + 1] != '<')

                                                                        {
                                                                        cadena[k] =
Final[l + j + 1];

                                                                                k++;
                                                                                l++;

                                                                        }
```

```
                                                        fin = 1;
                                                    }
                                                }
                                            }
                                        }
                                    }
                                }

        return cadena;
}
```