

# Práctica obligatoria 1

## El escáner de documentos

El siguiente enunciado corresponde a la primera práctica obligatoria de la asignatura Visión Artificial, de 4º curso en el Grado en Ingeniería Informática de la URJC.

### 1 Normas

Las prácticas se realizarán en grupos de 3 alumnos como máximo y se presentará usando el aula virtual.

La fecha límite de presentación es el día del examen será el domingo 30 de Marzo a las 23:00.

Para presentarla se deberá entregar un único fichero ZIP que contendrá el código fuente y un fichero PDF con la descripción del sistema desarrollado. Dicho fichero PDF incluirá una explicación con el algoritmo desarrollado, los parámetros utilizados y los resultados obtenidos. los métodos de OpenCv utilizados, copias de las pantallas correspondientes a la ejecución del programa y estadísticas correspondientes al resultado de la ejecución del programa sobre la muestra de test.

La puntuación de esta práctica corresponde al 30% de la asignatura. En particular se valorará:

- El correcto funcionamiento del algoritmo en al menos un 25% de las imágenes de test.
- El uso de las técnicas explicadas en clase para solucionar el problema.
- La división en funciones o clases del programa realizado.
- El exposición en el documento PDF del algoritmo desarrollado y de los resultados obtenidos.

La práctica deberá ejecutarse sobre Python 3.12 y OpenCV 4.11. y al menos consistirá en un fichero de python que se llamará scaner.py

Para ejecutar la práctica deberá escribirse en la consola de comandos “python” seguido del nombre de un fichero de tipo JPG, sin ningún otro parámetro adicional. Al ejecutar este comando se mostrará por pantalla el resultado sobre la imagen suministrada.

El código desarrollado en las prácticas debe de ser original. La copia (total o parcial) de prácticas será sancionada, al menos, con el SUSPENSO global de la asignatura en la convocatoria correspondiente. En estos casos, además, no regirá la liberación de partes de la asignatura (habrá que volver a presentarse al examen) y podrá significar, en la siguiente convocatoria y a discreción del profesor, el tener que **resolver nuevas pruebas y la defensa de las mismas de forma oral. Las sanciones derivadas de la copia, afectarán tanto al alumno que copia como al alumno copiado.**

Para evitar que **cuando se usa código de terceros** sea considerado una copia, se debe **citar siempre la procedencia** de cada parte de código no desarrollada por el propio alumno (con comentarios en el propio código y con mención expresa en la memoria de las referencias). El plagio o copia de terceros (p.ej. una página web) ya sea en el código a desarrollar en las prácticas y/o de parte de la memoria de

las prácticas, sin la cita correspondiente, acarrearán las mismas sanciones que en la copia prácticas de otros alumnos.

## 2 Introducción

Se desea construir una aplicación en Python que permita obtener una versión rectificada de una hoja de papel contenida en una fotografía digital. Tal aplicación permitiría obtener el mismo resultado que se obtiene al escanear una hoja de papel utilizando una fotografía de la cámara de un smartphone. La Figura 1 muestra un ejemplo de la entrada que recibiría la aplicación, y la salida que se esperaría de ésta.

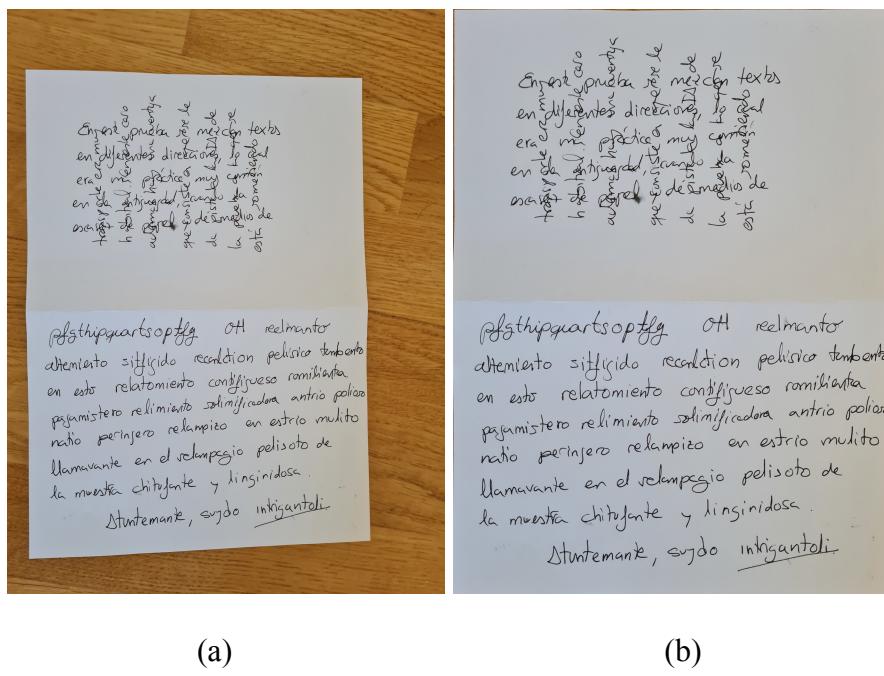
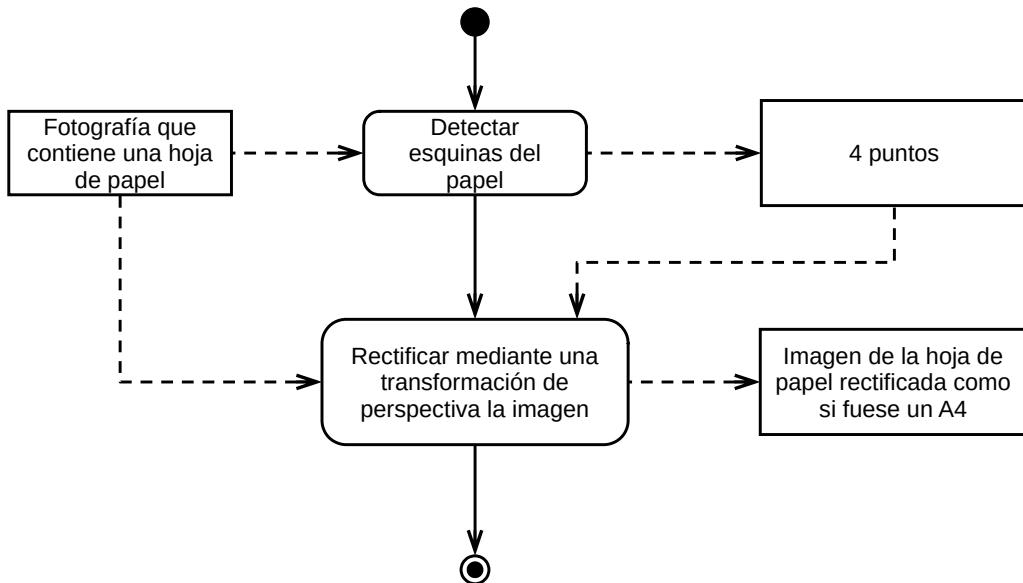


Figura 1.- Ejemplo de la entrada (a) y la salida deseada (b) para la aplicación del escáner de documentos.

Para construir dicha aplicación, se propone un algoritmo con dos pasos principales:

- Localización de las esquinas del papel
- Rectificación de la hoja.

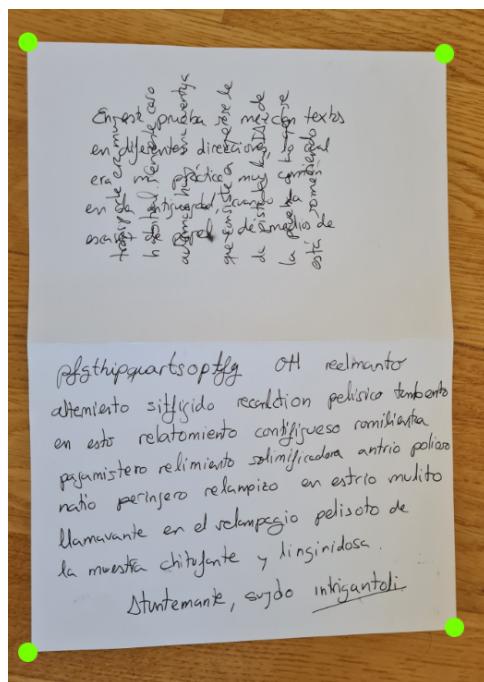
El diagrama de Actividad UML de la Figura 2 resume el funcionamiento planteado. En las siguientes secciones de este documento se describe en detalle cómo construir un sistema como el solicitado.



*Figura 2.- Diagrama UML de actividad con el esquema de funcionamiento planteado.*

### 3 Detectar esquinas del papel

En esta sección, se describe un proceso para construir una función que, partiendo de la fotografía digital de una hoja de papel sobre una superficie plana arbitraria, devuelva la posición de las cuatro esquinas que delimitan dicha hoja de papel (ver Figura 3).



*Figura 3.- Ejemplo de localización de las esquinas de una hoja de papel.*

### 3.1 Construcción de la función de detección

Para crear una función que detecte las esquinas de la hoja de papel se propone utilizar un enfoque basado en la detección de puntos singulares (Harris, FAST...), seguido de una descripción de los mismos (BRIEF, SIFT, etc.). Dicha descripción se utilizará para filtrar los puntos que no pertenezcan a las esquinas. El diagrama de la Figura 4 resume el proceso descrito. En dicho diagrama juega un papel central el knn-matcher que permite filtrar los puntos de interés que realmente pertenecen a las esquinas del papel del resto de puntos de interés que se puedan encontrar en la imagen.

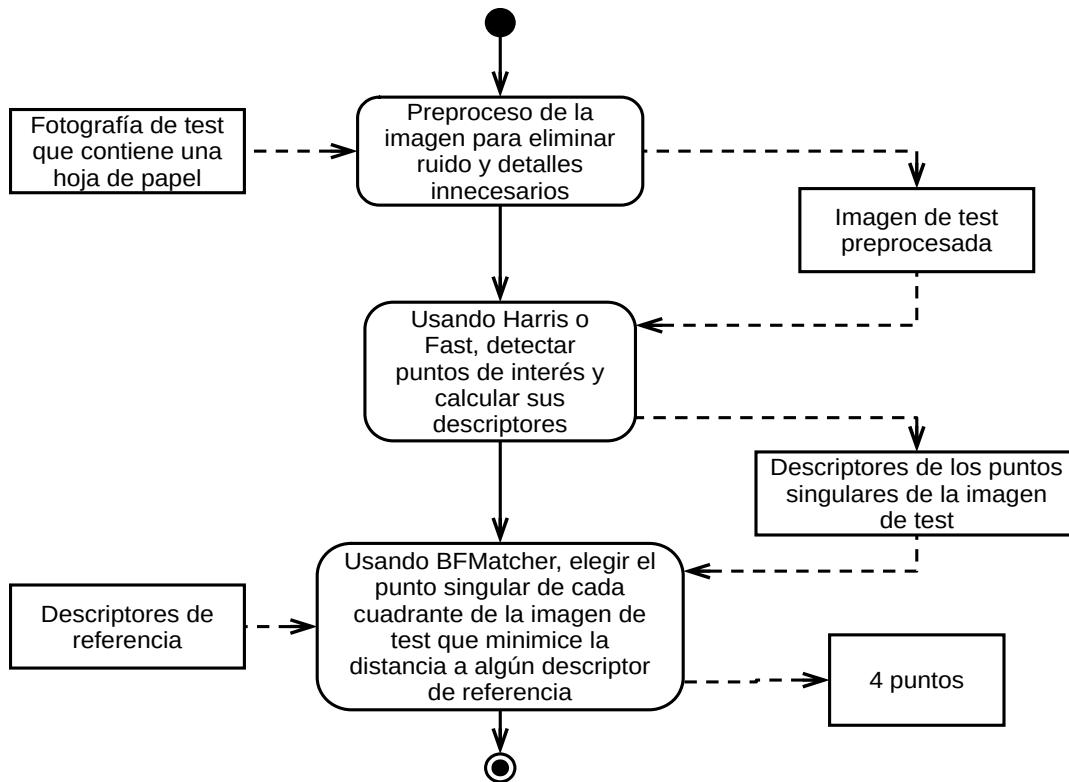


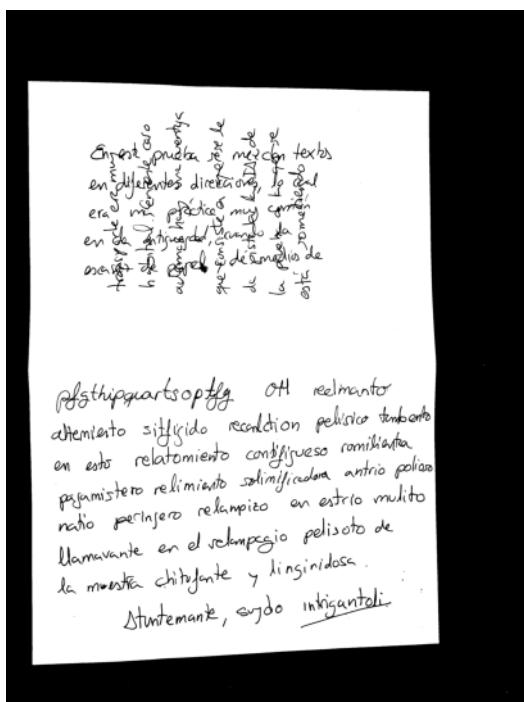
Figura 4.- Diagrama UML de actividad con el esquema de funcionamiento planteado.

Primeramente, se deberán almacenar los descriptores asociados a los puntos de las esquinas de los papeles de las imágenes de aprendizaje. Para ello, se recomienda seguir los siguientes pasos:

1. Utilizar un conjunto de imágenes de aprendizaje para configurar los parámetros de los algoritmos que se utilizarán. Utilizar un conjunto de imágenes de test para probar dichas configuraciones.
2. Etiquetar las esquinas de las imágenes de aprendizaje utilizando un programa que permita dicho proceso. Por ejemplo, la aplicación gratuita [VIA](#) permite el etiquetado de puntos dentro de una imagen y posteriormente exportar las coordenadas de dichos puntos a un fichero de texto.
3. Construir una función que, partiendo de las esquinas etiquetadas en el paso 2, calcule y almacene en una lista los descriptores de dichos puntos. En este caso recomendamos usar los descriptores de BRIEF (usando el método compute de un objeto BriefDescriptorExtractor). Los descriptores almacenados serán los descriptores de referencia.

Dentro del diagrama de la Figura 4, el primer paso consiste en el preprocessamiento de la imagen. Este preprocessamiento puede hacerse en dos fases: umbralizado y morfología.

1. Teniendo en cuenta que la mayor parte de la imagen corresponde a un papel de fondo blanco, se recomienda umbralizar la imagen antes de buscar los puntos singulares. Un proceso de umbralizado permitiría eliminar detalles innecesarios que dificultan la localización de los puntos singulares (ver Figura 5 de una imagen umbralizada). Para encontrar el valor umbral se pueden buscar picos en el histograma suavizado, por ejemplo usando las funciones `gaussian_filter1d` o `find_peaks` de `scipy`.
  2. Para eliminar ruido de la imagen bitonal, y facilitar aún más la tarea al proceso de detección de puntos singulares, puede ser útil utilizar operaciones morfológicas de erosión y dilatación.



*Figura 5.- Ejemplo de imagen preprocesada.*

Una vez que se tienen los descriptores de referencia y la imagen preprocesada, se puede probar a localizar las esquinas de un papel de la muestra de test. Para realizar dicha prueba se puede proceder en 3 pasos:

1. Obtener puntos de interés sobre las imágenes del conjunto de test. Para ello se deberá utilizar alguna una función que, para una imagen devuelva los puntos singulares que contiene. En este caso recomendamos usar Harris, aunque también se podría probar Fast. A los puntos detectados mediante Harris quizás se le puede aplicar un algoritmo de supresión de mínimos locales, para reducir su número.
  2. Calcular los descriptores asociados a los puntos singulares detectados en el paso 1.
  3. En cada cuadrante de la imagen de test, quedarse con el punto singular que tenga una distancia menor entre su descriptor y algún descriptor de referencia. Para calcular esta distancia entre descriptores se recomienda usar el método knnMatch de un objeto BFMatcher utilizando la

distancia de Hamming. De esta forma se obtendrán 4 puntos que deberían corresponder a las 4 esquinas del papel dentro de la imagen.

## 4 Recorte y rectificación del papel

Una vez que se dispone de los 4 puntos que corresponden a las esquinas de la hoja de papel, se desea desarrollar una función que rectifique la imagen inicial usando una transformación de perspectiva. Así, dicha función recibirá una imagen y devolverá la imagen rectificada de la hoja que contenga. En dicha transformación se considerará crear una imagen con un tamaño proporcional a un A4 y con un ancho similar a la distancia entre las dos esquinas superiores detectadas.

Finalmente, se deberá crear un fichero ejecutable desde python que ejecute todo el proceso recibiendo por parámetros el nombre del fichero a procesar y el nombre del fichero resultante.