UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola d'Enginyeria de Barcelona Est

TRABAJO DE FIN DE GRADO

**Grado en Ingeniería mecánica**

# TRATAMIENTO DIGITAL DE IMÁGENES EN MATLAB PARA LA DETECCIÓN DE FALLAS

## Memoria y anexos


**Autor:** Pablo Andrés Carela

**Director:** Pablo Buenestado Caballero

**Codirector**: Leonardo Acho Zuppa


**Convocatoria:** Junio 2019

# Resum:

El tractament digital d'imatges és una disciplina àmpliament desenvolupada i implementada en la vida quotidiana de la societat, prenent especial importància en sistemes de visió per computador en línies de producció industrials, així com en classificació d'imatges mèdiques o l'anàlisi de cultius mitjançant imatges aèries . La utilització d'aquests tipus de tecnologies ha propiciat un augment considerable de les eines de què disposen els professionals de diferents sectors.

Aquest projecte pretén trobar un mètode que permeti classificar automàticament diferents imatges de plaques solars per detectar falles superficials que puguin afectar l'estructura i al correcte funcionament de les mateixes. Per a això s'ha dut a terme un senzill procés de segmentació de les imatges, l'etiquetatge dels diferents objectes per tal de detectar el fons de la imatge i diferenciar-se clarament de la zona d'interès de la imatge, i, un cop acolorit tot , calcular el valor mitjà de la intensitat dels píxels, així com la variància per tenir dues variables de control que permetin fer una classificació més precisa. Finalment s'estudiarà el rendiment de l'algorisme duent a terme una classificació de diferents imatges preses a diferent resolució i amb diferents tipus de falla simulada.

**Paraules clau:** Imatge binària, mètodes estadístics, plaques solars, tractament d'imatges, augment de nitidesa, Matlab.

# Resumen:

El tratamiento digital de imágenes es una disciplina ampliamente desarrollada e implementada en la vida cotidiana de la sociedad, tomando especial importancia en sistemas de visión por computador en líneas de producción industriales, así como en clasificación de imágenes médicas o el análisis de cultivos mediante imágenes aéreas. La utilización de estos tipos de tecnologías ha propiciado un aumento considerable de las herramientas de que disponen los profesionales de diferentes sectores.

Este proyecto pretende encontrar un método que permita clasificar automáticamente diferentes imágenes de placas solares para detectar fallas superficiales que puedan afectar a la estructura y al correcto funcionamiento de las mismas. Para ello se ha llevado a cabo un sencillo proceso de segmentación de las imágenes, el etiquetado de los diferentes objetos con el fin de detectar el fondo de la imagen y diferenciarlo claramente de la zona de interés de la imagen, y, una vez coloreado todo, calcular el valor medio de la intensidad de los píxeles, así como la varianza para tener dos variables de control que permitan hacer una clasificación más precisa. Finalmente se estudiará del rendimiento del algoritmo llevando a cabo una clasificación de diferentes imágenes tomadas a diferente resolución y con diferentes tipos de falla simulada.

**Palabras clave:** Imagen binaria, métodos estadísticos, placas solares, tratamiento de imágenes, aumento de nitidez, Matlab.

# Abstract:

The digital processing of images is a discipline widely developed and implemented in the daily life of society, taking special importance in computer vision systems in industrial production lines, as well as in the classification of medical images or the analysis of agricultural crops using aerial images. The use of these types of technologies has led to a considerable increase in the tools available to professionals from different sectors.

This project aims to find a method to automatically classify different images of solar panels to detect surface faults that may affect the structure and the proper performance of these. To do this, it has been carried out a simple process of segmentation of the images, the labelling of the different objects in order to detect the background of the image and clearly differentiate it from the area of interest of the image, and, once coloured all, calculate the mean of the intensity of the pixels, as well as the variance to have two control variables that allow a more accurate classification

**Keywords:** Binary image, statistical methods, solar panels, image processing, sharpness, Matlab.

# Agradecimientos:

En primer lugar, quiero agradecer a los que han sido mis tutores del trabajo de fin de grado, los Profesores: Pablo Buenestado Caballero y Leonardo Acho Zuppa. Por introducirme en el campo del tratamiento de imágenes y por los consejos que me han transmitido a la hora de enfocar el proyecto y la redacción de la memoria.

Quiero agradecer a mis padres y a mis abuelos por apoyarme siempre y animarme a seguir estudiando en los momentos más difíciles de la carrera.

También quiero agradecer a mis compañeros de piso, Alberto y Rodrigo, que siempre han estado ahí, aguantándome en el día a día tanto en los momentos difíciles y aburridos, como en las noches de Mario Kart.

También les quiero agradecer a Bartomeu, Vicent y Victor que siempre tengan un buen tema de discusión en cualquier momento del día, a mis compañeros del bar, porque al café y la partida de guiñote no se puede faltar, a Jordi y a los colegas de Ancha es Castilla por estar dispuestos siempre a ir al bar y hacerme ir a clase cuando menos me apetecía. Y a mis amigos de Alcañiz por estar siempre dispuestos a salir de casa a cualquier hora.

Por último, no quiero olvidarme del CMU Ramón Llull y de la segunda madre que tenemos todos los llullatas en nuestra segunda casa. Gracias a Elena por la cantidad ingente de café y bocatas que nos preparáis las chicas y tu en el bar siempre que aparecemos por la puerta, así como por escucharnos y ayudarnos a adaptarnos a la vida fuera de casa.

# Índice:

# 1. Introducción:

Una imagen es una representación visual que manifiesta la apariencia de un objeto cualquiera real o imaginario. Existen diferentes técnicas para la representación de imágenes, algunas de ellas son el dibujo, la pintura o la fotografía. Las imágenes en su conjunto, representan una de las principales formas de transmitir información. En el presente proyecto el término imagen será referido a fotografías.

La fotografía es el arte o la técnica de obtener imágenes duraderas debido a la acción de la luz. Para poder llevar a cabo la captura de una fotografía es necesario un medio tanto químico, (fotografía analógica) como electrónico capaz de convertir la luz en señales electrónicas (fotografía digital). Independientemente de cómo hayan sido tomadas las fotografías, para poder realizar un tratamiento de las mismas por computadora es necesario que dichas fotografías estén digitalizadas. Durante el proceso de digitalización de una fotografía, un equipo electrónico lleva a cabo una discretización de la imagen, es decir, divide la imagen en pequeñas unidades de información, cada una de estas unidades almacena la información de la intensidad de cada uno de los tres colores primarios. A cada una de las unidades de información se les conoce como píxeles. El proceso de discretización es necesario puesto que los sistemas electrónicos digitales, como los ordenadores, codifican la información en estados discretos los cuales solo pueden tomar dos valores 0 o 1. De esta manera una imagen digital puede ser interpretada como una tabla donde cada celda guarda un valor numérico (fotografía en escala de grises, *ilustración 1 izquierda)* o un conjunto de 3 tablas donde cada una de las celdas guarda el valor un valor numérico de cada uno de los tres colores primarios (fotografía a color, *ilustración 1 derecha*) en ese punto determinado.

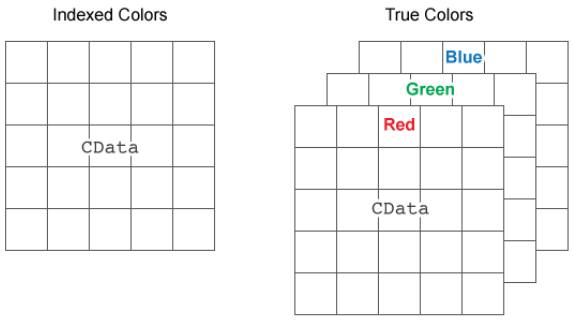


*Ilustración 1 - ejemplo de imagen digital (Fuente: bibliografía [6])*

Dependiendo del sistema empleado para la representación de la imagen se pueden clasificar en ***imágenes matriciales*** o en ***gráficos vectoriales.***

## 1.1. Imágenes matriciales:

Una imagen matricial o de mapa de bits es una estructura que representa una rejilla rectangular de píxeles o puntos de color, denominada matriz.

A las imágenes en mapa de *bits* se las define por su altura y su anchura, también denominado resolución, así como por su profundidad de bits, que determina el número total de colores que se pueden almacenar en cada uno de los puntos que forman la imagen, y, por lo tanto, en gran medida, la calidad del color de la imagen, es decir, la cantidad de bits que posee cada uno de los píxeles de la imagen.

Una imagen matricial normalmente es representada como ***I(x,y)*** donde el valor de la intensidad se obtiene del indexado de las coordenadas x e y (*Ecuación 1*).

$$I(x,y) = \begin{pmatrix} I(1,1) & \cdots & I(N,1) \\ \vdots & \ddots & \vdots \\ I(1,M) & \cdots & I(N,M) \end{pmatrix}$$

## 1.2. Imágenes vectoriales:

Una imagen vectorial imagen vectorial está formada por objetos geométricos definido cada uno de ellos por sus atributos matemáticos, por ejemplo, un círculo rojo quedaría definido por la posición de su centro, su radio, el grosor de línea y el color. Cuya principal aplicación es poder ampliar y reducir el tamaño de la imagen a voluntad sin perder calidad (*ilustración 2*).

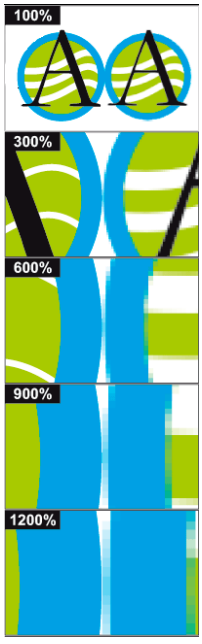


*Ilustración 2 - Diferencia entre imágenes matriciales y vectoriales (Fuente: bibliografía [8])*

## 1.3. Panel solar y fallas:

En el presente proyecto van a ser analizadas una serie de imágenes de paneles solares, algunos de los cuales se encontrarán en perfectas condiciones y en otros se habrán recreado fallos superficiales.

La radiación solar que alcanza la Tierra ha sido aprovechada por el ser humano desde la antigüedad, mediante diferentes tecnologías. Hoy en día, el calor y la luz del sol puede aprovecharse por medio de diversos captadores, conocidos como **paneles solares**. El aprovechamiento de la energía proveniente del sol, o energía solar, es una de las llamadas energías renovables.

Un **panel solar** es un dispositivo de captación de energía proporcionada por la radiación solar. Este término comprende tanto a los captadores solares, utilizados usualmente para producir agua caliente como a los paneles fotovoltaicos (*ilustración 3*), los cuales son utilizados para generar electricidad mediante el efecto fotoeléctrico.

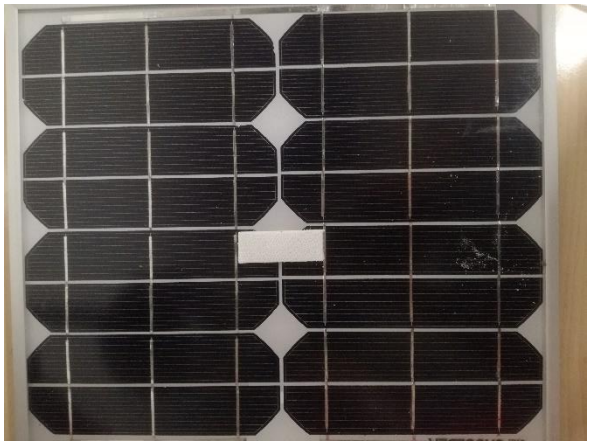

*Ilustración 3 - Ejemplo de panel fotovoltaico (Fuente: elaboración propia)*

Las fallas superficiales que han sido recreadas son principalmente de tres tipos: grietas (*Ilustración 4 izquierda*), polvo (*Ilustración 4 centro*) u obstrucción parcial del panel (*Ilustración 4 derecha*).

*Ilustración 4 - Ejemplos de fallas utilizadas (Fuente: elaboración propia)*

## 1.4. Estado del arte:

El procesamiento digital de imágenes puede definirse como la operación sobre imágenes mediante el uso de ordenadores con el objetivo de imitar, hasta cierto punto, el proceso de visión biológico.

Un sistema de visón y procesamiento de imágenes se compone de una serie de subsistemas que operan sobre una imagen con el objetivo de interpretar alguna característica notable.

A diferencia de los procesos de visión por computador, en los procesos de tratamiento de imágenes las imágenes de estudio pueden provenir tanto del espectro visible como de diferentes áreas del espectro electromagnético.

Hoy en día, prácticamente no existe ninguna área de la vida cotidiana en la que no se apliquen de una forma u otra algún tipo de tratamiento digital de imágenes. Desde procesos automatizados a gran escala hasta una simple publicación en una red social. El tratamiento digital de imágenes forma parte de la vida actual en mayor o menor medida.

Para cualquier proceso industrial es fundamental detectar las fallas que pueda tener un elemento de una cadena productiva, tanto de los elementos producidos como de los elementos productores, con el fin de minimizar las pérdidas, tanto económicas como logísticas, o en el caso de estudio del presente proyecto, maximizar la energía producida por los paneles solares

En el presente proyecto se va a desarrollar un algoritmo de detección de fallas en paneles solares mediante el uso de Matlab como interfaz de desarrollo. La principal ventaja del uso del tratamiento de imágenes es la rapidez con las que se analizan grandes cantidades de imágenes, reduciendo el tiempo en que se tarda en detectar una falla y, por lo tanto, maximizar la producción de energía de una instalación solar fotovoltaica.

# 2. Objetivo:

## 2.1. Especificaciones y requerimientos:

El objetivo principal de este trabajo es encontrar un método de clasificación de imágenes de placas solares para detectar posibles fallas visuales de manera automática con diferentes resoluciones, esto es debido a que cualquier equipo capaz de realizar una fotografía podría ser utilizable, desde cámaras de fotos profesionales hasta cámaras web fijas de baja resolución pasando por teléfonos móviles. Esta versatilidad a la hora de la toma de las imágenes permite al usuario del algoritmo reducir costes, ya que podría llevar a cabo dicha toma con cualquier equipo que tuviera a mano y, por lo tanto, facilitar el uso del mismo a prácticamente cualquier usuario que supiera utilizar un teléfono y copiar las imágenes a una carpeta en un ordenador. También permite realizar las tomas en cualquier momento sin la necesidad de depender de equipos fijos o de personal externo, cualquier propietario de una instalación solar fotovoltaica podría controlar directamente sus propias placas con su propio equipo fácilmente.

Las tareas necesarias para lograr el objetivo del proyecto son las siguientes:

- **Obtención del conjunto y preprocesamiento de las imágenes**: Adquisición, clasificación de las imágenes según el tipo de placas solares y separación de imágenes en patrones e imágenes a analizar.

- **Obtención de los valores estadísticos de las imágenes**: En primer lugar, se analizarán una serie de imágenes patrón de las placas en perfecto estado con diferentes orientaciones e iluminaciones. Para el análisis de las imágenes se realizará una segmentación, etiquetado y coloreado de las mismas. Una vez coloreadas las imágenes etiquetadas se procederá a calcular la media y la varianza de las imágenes de etiquetas.

- **Clasificación de las imágenes:** Una vez obtenidos los estadísticos de los patrones se procederá a realizar el mismo proceso con las imágenes de prueba. Una vez obtenidos los estadísticos se compararán con los valores obtenidos de los patrones. Si las imágenes se salen del rango formado por los patrones, las imágenes con posibles fallas se almacenarán en una carpeta a parte de las carpetas originales.

## 2.2. Entorno de programación utilizado:

Para el desarrollo del algoritmo de clasificación se ha utilizado MATLAB como lenguaje de programación. MATLAB posee un gran número de herramientas preprogramadas que permiten el manejo y uso de imágenes de manera rápida y eficiente, sin ser necesario programar desde 0 las funciones necesarias para el análisis de las imágenes y las diferentes operaciones que se llevan a cabo con las mismas. MATLAB también cuenta con numerosas librerías y paquetes de funciones, así como una documentación bastante amplia y con numerosos ejemplos de aplicación que facilitan el aprendizaje y la implementación de las diferentes funciones utilizadas.

# 3. Metodología:

Tal y como se expone en el capítulo 2, el funcionamiento del algoritmo se puede dividir a grandes rasgos en tres procesos claramente diferenciados: bloque1, bloque 2 y bloque 3. Por ello su desarrollo teórico y aplicación va a ser expuesta en tres apartados ordenados según el orden en el que se aplican los diferentes elementos en el algoritmo.

A pesar de dividir la metodología del algoritmo en bloques, existen una serie de conceptos teóricos, como la vecindad o la conectividad, que debe de ser enunciados previamente al desarrollo de los diferentes bloques, puesto que son necesarios para el desarrollo de algunos de los conceptos teóricos desarrollados dentro de los bloques.

## 3.1. Vecindad:

La vecindad se define como la relación que tiene un pixel con los píxeles más cercanos a él. Existen dos tipos de vecindad que posee un pixel en una imagen, la 4-vecinos y la 8-vecinos.

- **4-vecinos:** La vecindad 4-vecinos se constituye de los píxeles V1, V2, V3 y V4, que se encuentran encima y debajo, así como a la izquierda y la derecha del pixel P en cuestión.

| | | |
|---|---|---|
| | V2 | |
| V1 | P | V3 |
| | V4 | |

*Ilustración 5 - Ejemplo de 4-vecindad (Fuente Propia)*

- **8-vecinos:** La vecindad 8-vecinos se constituye con los pixeles V1, V2, V3, V4, V5, V6, V7 y V8. Los píxeles correspondientes a los 4-vecinos, más los que se encuentran en forma diagonal al pixel P en cuestión.

| | | |
|---|---|---|
| V2 | V3 | V4 |
| V1 | P | V5 |
| V8 | V7 | V6 |

*Ilustración 6 - Ejemplo de vecindad 8-vecinos (Fuente Propia)*

La vecindad es un concepto de suma importancia en el tratamiento digital de imágenes puesto que la mayoría de operaciones que se llevan a cabo durante el procesado de la imagen sigue criterios de vecindad de un pixel con sus 4 u 8 vecinos.

## 3.2. Conectividad:

La conectividad entre píxeles es un concepto utilizado ampliamente en la detección de regiones u objetos presentes en una determinada imagen. Por esta razón la conectividad se define como la situación de vecindad, y al igual que con la vecindad, existen dos tipos de conectividad: la conectividad a 4 y la conectividad a 8, siguiendo los mismos criterios que en la vecindad. Por lo tanto, se dice que dos píxeles, con el mismo valor de intensidad, están conectados con conectividad-4 o conectividad-8 si se encuentran en situación de 4-vecindad u 8-vecindad.

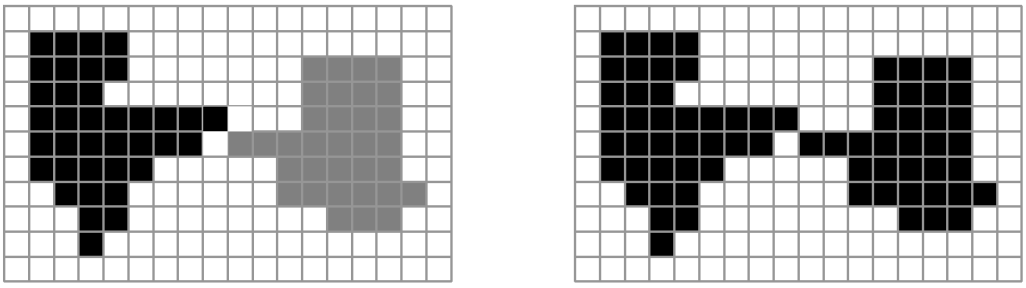

*Ilustración 7 - Ejemplos de conectividad-4 (izquierda) y de conectividad-8 (derecha) (Fuente Bibliográfica [9])*

## 3.3. Operaciones de pixel:

Las operaciones de pixel son aquellas que, realizadas sobre toda la imagen, solo se tiene en cuenta el valor del pixel en cuestión. Cada nuevo valor del pixel calculado, ***p' = I'(x,y)***, es dependiente del valor del pixel original ***p = I(x,y)*** en la misma posición y, con ello, independiente de los valores de otros píxeles. El nuevo valor del pixel es determinado a través de una función ***f[I(x,y)]***, es decir:

$$f[I(x,y)] \rightarrow I'(x,y)$$

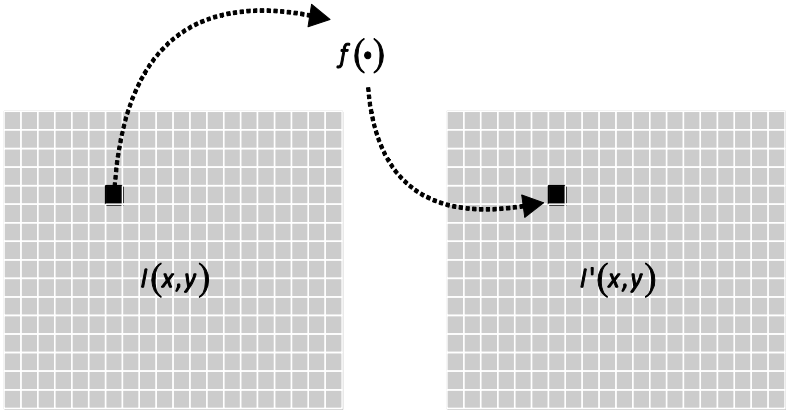


*Ilustración 8 - Representación de las operaciones de pixel (Fuente Bibliografía [9]])*

Algunos ejemplos de operaciones de píxel típicas llevadas a cabo en el presente proyecto son:

- Cambios de contraste e iluminación.

*Ilustración 9 - Ejemplo de aumento de iluminación en una imagen (original izquierda, modificada derecha) (Fuente: elaboración propia)*

- Segmentación por umbral de una imagen

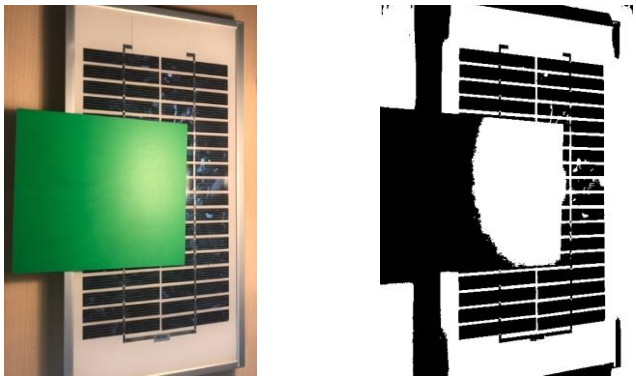

*Ilustración 10 - Ejemplo de segmentación de una imagen (original izquierda, imagen segmentada derecha) (Fuente: elaboración propia)*

## 3.4. Imagen binaria:

Las imágenes binarias son imágenes matriciales en las cuales los pixeles solo tienen dos valores: 0 o 1. Por lo general, esta clasificación corresponde a objetos presentes en la imagen (valores 1) y el fondo de la imagen (valores 0), aunque esta diferenciación en imágenes reales no siempre es posible. Un ejemplo de imagen binaria se puede apreciar en la *Ilustración 10 derecha*.

## 3.5. Bloque 1:

En el primer bloque del programa, se procede a cargar las imágenes en el programa y a realizar el preprocesado de las mismas. Para ello, en primer lugar, es necesario clasificar manualmente las imágenes de acuerdo con un criterio establecido; en segundo lugar, las imágenes deben de ser cargadas en el programa desde la carpeta que las contenga; y en tercer lugar se debe de aplicar el tratamiento necesario para la posterior aplicación de las operaciones contenidas en el bloque 2, que son las encargadas de detectar si una imagen contiene fallas.

Las etapas contenidas dentro del bloque 1 son las siguientes:

- Clasificación de las imágenes según tipos.
- Carga del conjunto de imágenes en el programa.
- Preprocesado de las imágenes.

### 3.5.1. Clasificación de las imágenes según tipos:

El conjunto de las imágenes que se han empleado procede de dos paneles solares del CODAlab en el EEBE. Todas las imágenes han sido tomadas con teléfono móvil a diferentes resoluciones y a pulso, es decir, sin ningún tipo de trípode o sistema de estabilización, con el fin de tener un abanico de estudio amplio a la hora de obtener los valores estadísticos.

La batería de imágenes sobre las cuales se va a trabajar a lo largo de todo el proyecto responde a la siguiente clasificación:

*Tabla 1 - Clasificación de las imágenes patrón*

| Tipo de imágenes | Resolución de las imágenes | | | Imágenes Totales |
|---|---|---|---|---|
| | 480x640 | 1200x1600 | 4160x3120 | |
| Tipo 1 | 4 | 0 | 14 | 18 |
| Tipo 2 | 2 | 0 | 8 | 10 |
| Tipo 3 | 0 | 0 | 4 | 4 |

*Tabla 2 - Clasificación de las imágenes que van a ser analizadas*

| Tipo de imágenes | Resolución de las imágenes | | | Imágenes Totales |
|---|---|---|---|---|
| | 480x640 | 1200x1600 | 4160x3120 | |
| Tipo 1 | 8 | 14 | 21 | 43 |
| Tipo 2 | 6 | 15 | 20 | 41 |
| Tipo 3 | 0 | 1 | 7 | 8 |

La clasificación de los tipos de imágenes corresponde a un criterio de construcción. Puesto que se han analizado imágenes de dos placas completamente diferentes entre ellas, se decidió separarlas en diferentes lotes según la construcción de la placa. Una vez separadas por placa se ha procedido a analizarlas según los diferentes tamaños, esto es debido a que a la hora de evaluar el rendimiento del algoritmo se llevan a cabo diferentes pruebas en las cuales la resolución de las imágenes es una variable.

Tal y como se ha mencionado anteriormente, el algoritmo desarrollado se ha probado con 3 tipos de imágenes completamente diferentes según un criterio de construcción de la placa. Los 3 tipos de imágenes son los siguientes:

- **Imágenes de tipo 1:**

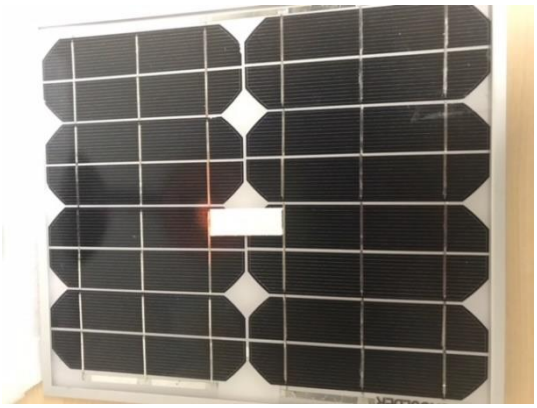

*Ilustración 11 - Ejemplo de imagen de tipo 1 (Fuente Propia)*

- **Imágenes de tipo 2:**

*Ilustración 12 - Ejemplo de imagen de tipo 2 (Fuente Propia)*

- **Imágenes de tipo 3:**

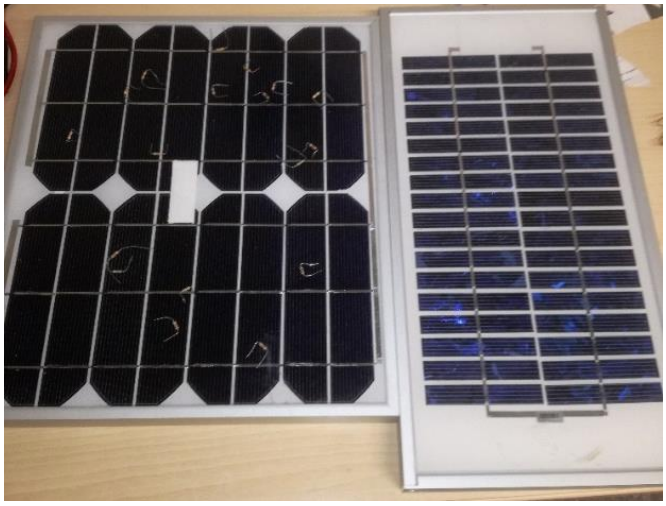

*Ilustración 13 - Ejemplo de imagen de tipo 3*

### 3.5.2. Carga del conjunto de imágenes en el programa:

Una vez llevada a cabo la clasificación de las imágenes en diferentes tipos de acuerdo a la construcción de las mismas, debido a que la función *imread* (función para importar una imagen) solo puede cargar imágenes de una en un se ha de buscar la manera de cargar la batería de imágenes al completo en el programa. Para poder trabajar cómodamente con todas las imágenes sin tener la necesidad de importar una a una de forma manual se ha optado por utilizar un *Datastore*.

Un *Datastore* en MATLAB es una variable, la cual se utiliza para poder manejar una colección entera de datos directamente desde la carpeta de origen y acceder a ellos utilizando índices, puesto que los elementos que componen un *Datastore* se almacenan en forma de lista. Para el presente proyecto se ha utilizado en concreto un tipo de *Datastore* específico para imágenes:

```
imds = imageDatastore(location)
```

Donde:

- **Imds:** es la variable de tipo *ImageDatastore* creada
- **Location:** es la carpeta en la que se encuentran las imágenes que se quieren cargar al programa.

```
pruebas =

  ImageDatastore with properties:

                    Files: {
                           ' ...\Usuario\Uni\Q10\TFG\desarrollo\fotos\tipo 1\IMG-20190405-WA0018.jpg';
                           ' ...\Usuario\Uni\Q10\TFG\desarrollo\fotos\tipo 1\IMG-20190405-WA0019.jpg';
                           ' ...\Usuario\Uni\Q10\TFG\desarrollo\fotos\tipo 1\IMG-20190405-WA0022.jpg'
                            ... and 41 more
                           }
    AlternateFileSystemRoots: {}
                    ReadSize: 1
                      Labels: {}
                    ReadFcn: @readDatastoreImage
```

*Ilustración 14 - Ejemplo de un ImageDatastore (Fuente Propia)*

Como se puede observar en la *ilustración 8*, los datos que almacena el *ImageDatastore* son los siguientes:

- **Files:** son los archivos que almacena el *ImageDatastore*, se almacenan como un vector de caracteres, donde cada uno de los elementos del vector es la dirección completa donde se encuentran las imágenes.
- **AlternateFileSystemRoots:** almacena la dirección de la carpeta en la que se encuentran los archivos en el caso de que se esté trabajado con una maquina en línea con otro sistema operativo. En este proyecto no será utilizado puesto que todo el programa se desarrolla en la máquina local
- **ReadSize:** indica el número de imágenes que serán leídas por el programa cada vez que se utiliza cualquier tipo de función *read* como *imread*. Por defecto es 1
- **Labels:** En el caso de especificar un vector con los nombres que se les darían a las imágenes cargadas dentro del programa, después de utilizar una función *read*.
- **ReadFCN:** almacena el nombre de la función que se utilizará para cargar todas las imágenes una a una en el programa. Por defecto el valor de *ReadFCN* será *@readDatastoreImage*. En este proyecto no será utilizada ya que se cargarán las imágenes una a una.

En este proyecto dos *ImageDatastore* creados son los siguientes:

```
patrones = imageDatastore(carp_patr);
 pruebas = imageDatastore(carp_pru);
```

Donde *carp_patr* y *carp_pru* son dos variables que almacenan las direcciones de las carpetas donde se encuentran las imágenes. En este proyecto las carpetas son las siguientes:

```
carp_patr = 'fotos/tipo 1/patrones';
     carp_pru = 'fotos/tipo 1';
```

La estructura de carpetas, dentro de la capeta principal del proyecto, sobre la que se ha trabajado responde a la siguiente jerarquía:

- **fotos/tipo x**: es la carpeta principal del tipo estudiado, será donde se encuentren las imágenes de prueba que deban de ser clasificadas por el algoritmo.
- **fotos/tipo x/patrones:** en esta carpeta únicamente deberán estar aquellas imágenes que se consideren patrones.
- **fotos/tipo x/fallas:** será la carpeta donde el algoritmo almacene las imágenes que considere que contienen una falla. Esta carpeta debe de ser creada por el usuario antes de ejecutar el programa.

Una vez las imágenes estén cargadas en el espacio de trabajo, para que una función de MATLAB pueda acceder a ellas se deberá escribir el siguiente comando para indicar el elemento que debe de leer:

```
Imds.Files{i}
```

Donde **i** es un valor numérico entero y positivo que equivale a la posición ordinal que ocupa la imagen a la que se quiere acceder.

### 3.5.3. Preprocesado de las imágenes:

En el presente proyecto han sido aplicadas dos operaciones previas al análisis y detección de las fallas. Estas operaciones han sido:

- **Reescalado:** a la hora de evaluar el rendimiento del algoritmo una de las operaciones que se han practicado en el preprocesado ha consistido en ajustar el tamaño de las imágenes a la resolución más baja de todas las que presentan las imágenes de la batería, es decir, 480x640 píxeles. Para ello se ha utilizado la siguiente función de MATLAB:

  ```
  I = imresize(I,[480,640]);
  ```

  Donde **I** es la imagen que se quiere modificar

- **Aumento de la nitidez:** la operación de aumento de nitidez se ha aplicado a continuación del reescalado mediante la siguiente función de MATLAB:

  ```
  B = locallapfilt(A,sigma,alpha)
  ```

Donde B es la imagen mejorada, A es la imagen original, sigma caracteriza la amplitud de los bordes de la imagen tomando valores entre [0,1] y alpha controla el suavizado de los detalles.

Para el suavizado de los detalles el valor alpha puede estar entre [0.01,10] con los siguientes resultados:

- **Alpha < 1:** Aumenta los detalles de la imagen de entrada, mejorando eficazmente el contraste local de la imagen sin afectar los bordes ni introduciendo halos.
- **Alpha = 1**: Los detalles de la imagen de entrada se dejan sin cambios.
- **Alpha > 1**: Suaviza los detalles de la imagen de entrada mientras conserva los bordes nítidos

Para decidir el valor de los parámetros de la función *locallapfilt* con el objetivo de aumentar el nivel de detalle de las imágenes han sido obtenidos de manera experimental. Para ello se ha utilizado una imagen de ejemplo de cada uno de los tamaños y tipos y se les han aplicado los mismos parámetros a todas y se han evaluado los resultados visualmente (*ilustración 15*).

*Ilustración 15 - Imagen original (izquierda), imagen con la nitidez aumentada(derecha) (Fuente: elaboración propia)*

El procedimiento iterativo aplicado ha sido el siguiente:

```
for i = 1:n
        I = imread(pruebas.Files{i});
        I1 = locallapfilt(I,1,0.65);
        figure()
        subplot(1,2,1);
        imshow(I);
        subplot(1,2,2);
        imshow(I1);
end
```

Una vez se obtuvieron unos resultados satisfactorios se fijaron los valores y se aplicó la función *locallapfilt* dentro del bucle principal del algoritmo.

Para aumentar la nitidez la función *locallapfilt* lleva a cabo, en primer lugar, una detección de los bordes de la imagen, y en segundo lugar aplica un filtro laplaciano.

#### 3.5.3.1. Filtro:

En una imagen se observan regiones o píxeles en los que de manera local se da un cambio abrupto del nivel de intensidad: puede aumentar o disminuir drásticamente. En algunas situaciones se desea realizar un suavizado del nivel de intensidad en dicha región.

Una primera idea de suavizado es en la que cada pixel es reemplazado por el promedio de sus vecinos. Por lo que para calcular el valor del pixel de la imagen suavizada ***I'(x,y)*** se emplea el de la imagen original ***I(x,y) = $p_0$*** más los 8 píxeles vecinos ***$p_1$,…,$p_8$*** y se calcula el promedio aritmético de los 9 valores:

$$I'(x,y) = \frac{1}{9}\sum_{j=-1}^{1}\sum_{i=-1}^{1} I(x+i, y+j)$$

Este promedio calculado muestra todos los elementos típicos presentes en un filtro. Realmente lo enunciado anteriormente es el ejemplo típico de los llamados **filtros lineales**.

La forma de la región del filtro puede ser cualquiera. Sin embargo, la cuadrada es la más utilizada porque, además de la facilidad de cálculo, permite considerar los píxeles vecinos en todas direcciones, lo cual, es una propiedad deseable en un filtro.

También puede observarse la existencia de filtros que tienen características no lineales, que de forma implícita imponen ciertas propiedades a las operaciones que de ellos dependen, como el filtro laplaciano, el cual es utilizado por la función *locallapfilt*.

### 3.5.3.2. Detección de bordes y aumento de nitidez:

Como paso previo al aumento de nitidez la función *locallapfilt* realiza una detección de los bordes debido a que algunas características en una imagen tales como los bordes y contornos son detectados a través de cambios locales de intensidad o de color. Los bordes y contornos desempeñan un papel importante en la interpretación de imágenes. La subjetiva "claridad" de una imagen se encuentra en relación directa con las discontinuidades y nitidez de las estructuras presentes en la misma.

Los bordes tienen un predominante rol en la visión humana, no solamente son detectables, sino que también permiten reconstruir objetos mediante unas pocas líneas. Los bordes, grosso *modo*, pueden ser considerados como puntos en la imagen, en los cuales la intensidad en una determinada dirección cambia de manera drástica. El valor del borde en la imagen para dicho punto depende directamente del cambio de valor. El "tamaño" del cambio de valor es calculado normalmente con la derivada de la imagen en el punto en cuestión y es utilizada como uno de los principales métodos de detección de bordes en una imagen.

La operación matemática utilizada para detectar el cambio brusco de intensidad en una región de una imagen es la derivada, y por consiguiente, el gradiente si se efectúa una derivada local. Esto es debido a que la derivada de una función en un punto nos indica el valor de la pendiente en dicho punto, por lo tanto, para buscar un cambio brusco en el valor de la intensidad es necesario detectar un cambio en la pendiente de la intensidad.

Tal y como se aprecia en la *ilustración 16* para una función cualquiera (***f(x) = -x² + 2)*** azul en la ilustración), si calculamos el valor de la derivada en un punto cualquiera y trazamos una recta tangente a dicho punto (***f'(x = 1) = 2*** ,rojo en la ilustración), obtenemos una recta con pendiente igual al valor de la derivada en dicho punto.

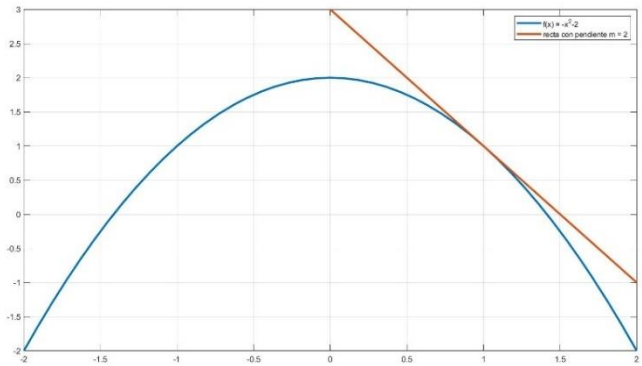


*Ilustración 16 - Ejemplo de derivada (Fuente: elaboración propia)*

Para el cálculo de la derivada se considerará una sola dimensión y se tomará como ejemplo una imagen que tenga una región blanca en el centro rodeada de un fondo oscuro.

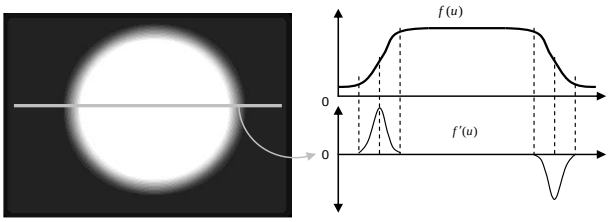


*Ilustración 17 - Imagen de ejemplo (derecha) y valor de la intensidad de la imagen y de su primera derivada (izquierda) (Fuente Bibliografía [9])*

Dicha imagen será considerada como una señal unidimensional ***f(u)*** y se calculará la primera derivada a partir de ella:

$$f'(u) = \frac{df}{du}(u)$$

Así, se produce una elevación positiva en todo lugar donde la intensidad aumente y una elevación negativa donde la intensidad disminuya. Sin embargo, la derivada es una operación matemática propia de funciones continuas, es decir, no está definida para funciones discretas como ***f(u)***, con lo que es necesario un método para calcularla. Para una función discreta, la derivada en un punto ***u*** puede ser calculada a partir de la diferencia existente entre los puntos vecinos a ***u***, dividido entre el valor de muestreo entre ambos puntos.

El valor de la derivada de una función discreta en un punto puede ser aproximada por:

$$\frac{df}{du}(u) \approx \frac{f(u+1) - f(u-1)}{2}$$

$$\frac{df}{du}(u) \approx 0{,}5 * \big(f(u+1) - f(u-1)\big)$$

Este mismo proceso puede ser replicado para los vecinos verticales del pixel en cuestión.

La **derivada parcial** puede ser considerada como la derivada de una función multidimensional a lo largo de un eje coordenado.

$$I(x,y) = \begin{bmatrix} \frac{\partial I}{\partial x}(x,y) \\ \frac{\partial I}{\partial y}(x,y) \end{bmatrix}$$

Sin embargo, igual que pasa con la derivada global de una función discreta, la derivada parcial no está definida. Una posible aproximación puede ser considerada como:

$$\frac{\partial f}{\partial x} = f(x+1) - f(x)$$

Para calcular el valor de la pendiente de la línea de los puntos vecinos *f(u-1)* y *f(u+1)* sirve como cálculo indirecto de la pendiente de la tangente en *f(u)*:

$$|\nabla I| = \sqrt{\left(\frac{\partial I}{\partial x}\right)^2 + \left(\frac{\partial I}{\partial y}\right)^2}$$

El valor del gradiente es invariante a rotaciones de la imagen y con ello también independiente de la orientación de las estructuras contenidas en la misma. Esta propiedad es importante para la localización de bordes en la imagen, es por esto que $|\nabla I|$ es el valor práctico utilizado en la mayoría de los algoritmos para la detección de bordes. Esta propiedad se conoce como **Isotropía**.

Si se aplican operaciones basadas en la primera derivada para la detección de bordes se obtiene el siguiente resultado:

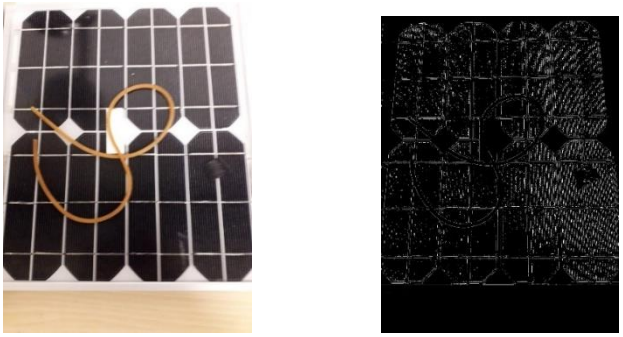

*Ilustración 18 -imagen original (derecha) y ejemplo de detección de bordes utilizando la primera derivada (derecha) (Fuente: elaboración propia)*

Las componentes del gradiente de las ecuaciones anteriores no son otra cosa que la primera derivada en el sentido de las columnas de la imagen. La forma de calcular la derivada en sentido horizontal es posible, de tal manera que se puede definir un filtro con la siguiente matriz de coeficientes:

$$H_x^D = 0{,}5 * \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}$$

Donde el coeficiente -0.5 afecta al pixel ***I(x-1 , y)*** *y* 0.5 al pixel ***I(x+1 , y)***. El valor del pixel central es multiplicado por 0, es decir, es ignorado.

De igual forma, se puede establecer el mismo filtro en sentido vertical, siendo su matriz de coeficientes:

$$H_y^D = 0{,}5 * \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}$$

El filtro del gradiente horizontal $H_x^D$ reacciona proporcionando una respuesta mayor en sentido horizontal y resaltando los bordes en sentido vertical. De igual manera, el filtro $H_y^D$ actúa e igual manera resaltando los bordes en sentido horizontal

A pesar de que el valor del gradiente permite detectar un borde, en ocasiones es importante tener información sobre si el pixel está en la transición positiva o negativa del gradiente. Para ello es necesario detectar los **pasos por cero**, es decir, los puntos, dentro del cambio brusco de intensidad, en los cuales la pendiente de la función ***f(u)*** es 0. La operación necesaria para detectar los pasos por cero es la segunda derivada.

Igual que pasaba con la primera derivada, la segunda derivada tampoco está definida para funciones discretas. Se puede definir la segunda derivada de una función discreta como:

$$\frac{\partial^2 f}{\partial x^2} = f(x+1) - f(x)$$

Tal y como se ha enunciado anteriormente, el valor del gradiente es isótropo. Por lo tanto, es deseable que los filtros basados en la segunda derivada conserven esta isotropía, es por ello que se utiliza el filtro laplaciano para llevar a cabo una mejora de la nitidez de la imagen.

El operador laplaciano, en el que está basado el filtro laplaciano, está definido por la siguiente expresión:

$$\nabla^2 I(x,y) = \frac{\partial^2 I(x,y)}{\partial x^2} + \frac{\partial^2 I(x,y)}{\partial y^2}$$

Si se lleva a cabo el desarrollo para la segunda derivada de funciones discretas de la ecuación anterior se obtiene:

$$\nabla^2 I(x,y) = I(x+1,y) + I(x-1,y) + I(x,y+1) + I(x,y-1) - 4 * I(x,y)$$

SI se expresa la ecuación anterior en términos de un filtro utilizable en el procesado de imágenes quedaría definido por la siguiente matriz:

| | | |
|---|---|---|
| 0 | 1 | 0 |
| 1 | -4 | 1 |
| 0 | 1 | 0 |

Si se aplica el operador laplaciano a una imagen se obtienen los bordes de los objetos presentes en la imagen. Sin embargo, si lo que se desea es mejorar la nitidez de la imagen, será necesario conservar la información de baja frecuencia de la imagen original y enfatizar los detalles presentes en la misma a través del filtro laplaciano. Para realizar este efecto, es necesario restar a la imagen original una versión escalada del valor del filtro laplaciano, por lo que la imagen con una nitidez mejorada quedaría definida de la siguiente manera:

$$I(x,y)_{Mejorada} = I(x,y) - (1) \cdot \nabla^2 I(x,y)$$

Y considerando que:

$$\nabla^2 I(x,y) = I(x+1,y) + I(x-1,y) + I(x,y+1) + I(x,y-1) - 4I(x,y)$$

Se obtiene:

$$I(x,y)_{Mejorada} = 5I(x,y) - [I(x+1,y) + I(x-1,y) + I(x,y-1) + I(x,y-1)]$$

O bien, expresado en un filtro, se obtiene:

$$I(x,y)_{Mejorada} = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

Gracias a la función *locallapfilt* no es necesario aplicar el desarrollo matemático presentado en este capítulo puesto que la función lo aplica todo automáticamente según los parámetros de nitidez que el usuario decida aplicar a la hora de utilizar la función.

## 3.6. Bloque 2:

Una vez cargadas las imágenes y preprocesadas, es el momento de extraer los valores estadísticos que serán utilizados para determinar si una imagen contiene posibles fallas. El proceso, que a continuación se expone, se realiza dos veces a lo largo de la ejecución del programa. Una primera vez para obtener los estadísticos de los patrones, y una segunda vez para obtener los estadísticos de las imágenes de prueba que va a clasificar el programa. Sin embargo, a pesar de que se va a llevar a cabo dos veces solo se va a exponer el desarrollo del mismo una  vezpuesto que el proceso se realiza exactamente igual tanto para los patrones como para las imágenes de prueba.

Igual que en el bloque 1, el bloque dos se puede dividir en diferentes etapas, que realizan diferentes procesos. Las etapas contenidas dentro del bloque 2 son las siguientes:

- Binarización de las imágenes.
- Etiquetado y detección de objetos.
- Obtención de los valores estadísticos de las imágenes.

El objetivo de binarizar, detectar los objetos y etiquetar las imágenes es conseguir homogeneizar el contenido de las mismas. El resultado de la homogeneización será el siguiente:

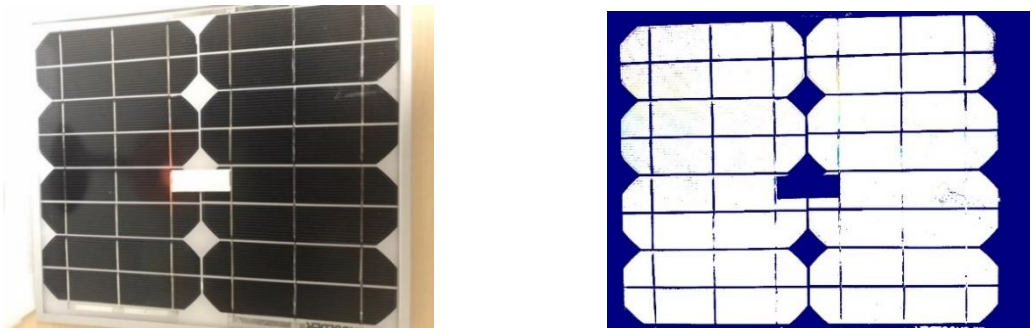

*Ilustración 19 – Imagen original (izquierda) y ejemplo de imagen homogeneizada (derecha) (Fuente: elaboración propia)*

Como se puede observar en la *ilustración 19*, se obtienen dos resultados claramente diferenciados. En primer lugar, obtenemos la estructura del panel solar en blanco donde las imperfecciones se mostrarían en azul, y en segundo lugar se obtiene el fondo de la imagen en azul. Este proceso permite eliminar el fondo de la imagen, obteniendo un color uniforme para todas las imágenes, lo cual permite obtener un rango de estadísticos equitativos para todas las imágenes independientemente del fondo de la mismas.

### 3.6.1. Binarización de las imágenes:

El primer paso para obtener imágenes binarias a partir de imágenes a color es separar cada uno de los colores en tres imágenes independientes, debido a que el proceso de binarización se lleva a cabo en imágenes en escala de grises. El motivo por el cual se ha decidido llevar a cabo esta separación en lugar de pasar las imágenes en color a imágenes en escala de grises es sencillo, si se separan las tres capas de una imagen a color en tres imágenes independientes se obtienen tres imágenes en escala de grises formadas por con los valores originales de la imagen. Al separar los colores se puede detectar cualquier falla que pudiera ser visible solamente en uno de los tres colores, utilizar la función *rgb2gray* de MATLAB perderíamos parte de la información puesto que el paso de color a escala de grises se lleva a cabo mediante una ponderación de los tres valores.

Para separar los tres canales se ha utilizado la siguiente función propia:

```
[I1RG,I1GG,I1BG] = separar_colores(I);
```

La cual devuelve tres imágenes formadas con la información proporcionada en la imagen **I** que se le pasa a la función como variable de entrada. El código de esta función se puede encontrar en el anexo A.

Una vez han sido separados las diferentes capas de las imágenes el siguiente paso es obtener las imágenes binarias correspondientes de las imágenes formadas por las capas de la imagen original. Para ello en MATLAB se utiliza la función:

```
BW = imbinarize(I)
```

Donde MATLAB devuelve una imagen binaria formada a partir de la información de la imagen **I**, para ello MATLAB lleva a cabo un proceso de segmentación de acuerdo con el valor de intensidad de los píxeles de la imagen.

#### 3.6.1.1. Segmentación:

La segmentación se define como el proceso de separar la imagen en diferentes regiones, de tal forma que los píxeles en cada región sean similares (de acuerdo con un determinado criterio). La idea principal es distinguir los objetos de interés del resto de la imagen. En el caso más simple, la segmentación es verificada considerando únicamente dos clases: la primera, en torno al objeto de interés, mientras que la segunda corresponde al fondo de la imagen. La segmentación debe ser considerada como un paso previo para la descripción, reconocimiento o clasificación de objetos contenidos en una imagen.

Existen muchos enfoques para la segmentación de imágenes, que se pueden dividir de acuerdo con las características de consideradas y con el tipo de algoritmo empleado. Los métodos de segmentación incluyen los valores de intensidad de los píxeles o la textura.

En cuanto al tipo de algoritmos que se emplean para llevar a cabo la segmentación de una imagen, se pueden dividir en ***no contextuales*** y ***contextuales***.

- Las **técnicas no contextuales** ignoran la relación existente entre los píxeles y el objeto que se intenta aislar. Los píxeles se agrupan considerando alguna característica global de ellos como el nivel de intensidad. Dentro de las principales técnicas de esta categoría se encuentra ***la umbralización***.

- Las **técnicas contextuales**, adicionalmente explotan la relación entre el pixel y el objeto que se pretende aislar mediante la incorporación de medidas o criterios

de homogeneidad. De esta manera, un método contextual podría agrupar un conjunto de píxeles que tienen una intensidad similar y que, al mismo tiempo, se encuentran cercanos entre sí, o que mantengan una misma dirección de acuerdo con su valor de gradiente.

La **umbralización** es la técnica utilizada por MATLAB para segmentar imágenes mediante la función *imbinarize*. La umbralización es una técnica de segmentación que parte del supuesto de que los objetos presentes en una imagen están formados de píxeles de intensidad homogénea. De esta manera, cada pixel es comparado con un umbral prefijado: si el valor del a intensidad del pixel es mayor, este es considerado de una determinada categoría, sin embargo, si el valor de la intensidad es menor, el pixel es de una región diferente. Bajo estas circunstancias, la calidad de la segmentación dependerá de la elección de un valor apropiado del umbral.

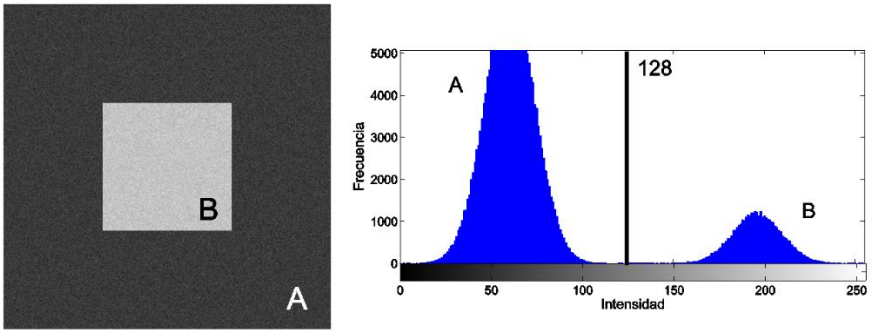


*Ilustración 20 - Distribución de las intensidades de los píxeles de dos objetos diferentes (Izquierda) histograma ideal resultante de la umbralización (Derecha) (Fuente Bibliografía [9])*

En la mayoría de las ocasiones, la distribución de los píxeles no mantiene una división tan clara como la mostrada en la Ilustración 8. Contrariamente, las distribuciones tienden a solaparse, lo cual determina la dificultad de la segmentación. Tal y como se muestra en la *ilustración 21*.

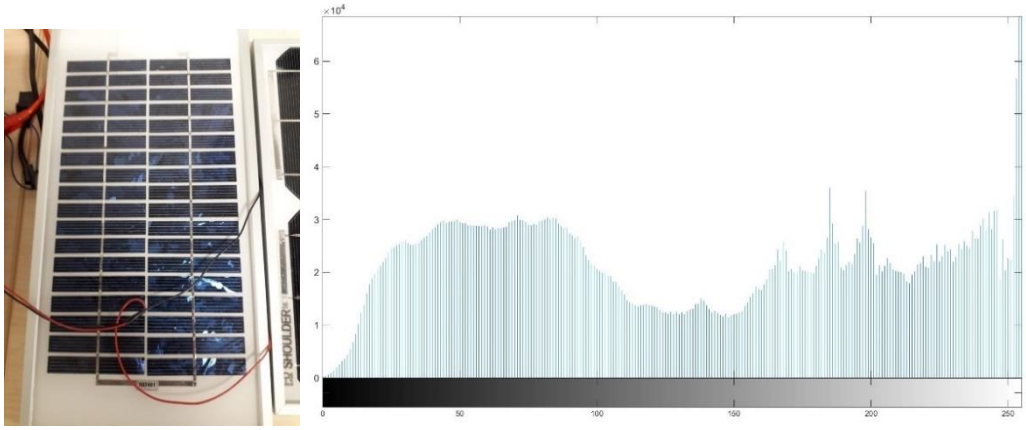

*Ilustración 21 - Imagen original (izquierda) y la distribución de su histograma (derecha) (Fuente: elaboración propia)*

En los casos que se produzca un solapamiento, no es posible encontrar un umbral que satisfaga una correcta segmentación. La incertidumbre provocada por el solapamiento de las distribuciones representa la causa de los posibles malos resultados de una segmentación. En estos casos de solapamiento, para poder llevar a cabo la segmentación es necesario calcular un **umbral óptimo**, gracias al cual se reduce la cantidad de píxeles mal clasificados, esto es: píxeles que pertenezcan al objeto A son clasificados como si pertenecieran al objeto B y viceversa. De esta manera, el umbral óptimo corresponde a la intersección entre ambas distribuciones.

Existen varios enfoques para la determinación del umbral óptimo en el solapamiento de distribuciones. La mayoría de ellos opera iterativamente, probando un determinado umbral, de tal forma que los píxeles de cada objeto estén más cercanos al valor promedio de sus respectivas intensidades, que al valor promedio de intensidad del objeto contrario. A la hora de calcular el valor del umbral óptimo MATLAB utiliza el **método Otsu** dentro de la función *imbinarize*.

El método **Otsu** es el más popular para el cálculo del umbral óptimo, ya que permite la binarización de dos clases usando el histograma de intensidades de la imagen. El método considera una imagen I de dimensiones MxN con L-1 escala de grises. El histograma presenta dos distribuciones solapadas: la primera corresponde al objeto A, involucra los valores de intensidad entre 0 y k, mientras que la segunda integra los niveles de gris de k+1 hasta L-1.

$$P(A) = \sum_{i=0}^{k} h(i) = w_A(k) = w(k)$$

$$P(B) = \sum_{i=k+1}^{L-1} h(i) = w_B(k) = 1 - w(k)$$

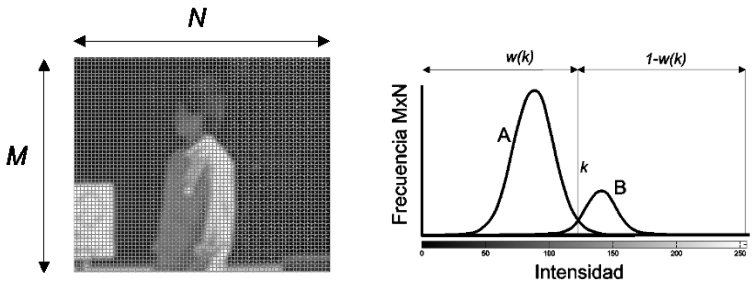


*Ilustración 22 - Proceso de segmentación de dos distribuciones mediante el método Otsu (Fuente Bibliografía [9])*

Donde **h( i )** representa el número de píxeles de intensidad *i* contenidos en la imagen. Bajo estas condiciones, los promedios de las intensidades de cada objeto pueden ser calculadas de la siguiente manera:

$$\mu_A(k) = \frac{1}{w_A(k)} \sum_{i=0}^{k} i \cdot h(i)$$

$$\mu_B(k) = \frac{1}{w_B(k)} \sum_{i=k+1}^{L-1} i \cdot h(i)$$

A partir de estos valores, las varianzas se pueden calcular de la siguiente manera:

$$\sigma_A^2 = \frac{1}{w_A(k)} \sum_{i=0}^{k} \big(i - \mu_A(k)\big)^2 \cdot h(i)$$

$$\sigma_B^2 = \frac{1}{w_B(k)} \sum_{i=0}^{k} \big(i - \mu_B(k)\big)^2 \cdot h(i)$$

Bajo estas circunstancias la varianza dentro de las clases puede definirse como:

$$\sigma_D^2 = w_A(k) \cdot \sigma_A^2(k) + w_B(k) \cdot \sigma_B^2(k)$$

Considerando esta información, el **método Otsu** busca como umbral óptimo, el nivel de intensidad k que minimiza la varianza existente dentro de las clases. De esta manera, la idea es minimizar la varianza de cada objeto con el fin de minimizar el solapamiento.

### 3.6.2. Etiquetado y detección de objetos:

El etiquetado de objetos es una técnica clásica del procesado de imágenes, consiste en recorrer una imagen binaria con el objetivo de detectar objetos diferentes en la imagen. El algoritmo lleva a cabo dos pasos consecutivos: en el primero se lleva a cabo un etiquetado temporal de los objetos y en el segundo las etiquetas son revisadas con el objetivo de detectar si dos etiquetas forman parte del mismo objeto o de objetos diferentes. El método es complejo, especialmente el segundo paso, sin embargo, la

moderada necesidad de memoria lo convierte en una buena opción para el etiquetado de objetos.

En el **primer paso,** En este primer paso la imagen es recorrida de derecha a izquierda y de arriba abajo. En cada recorrido se le asigna a cada pixel ( x , y ) una etiqueta temporal. El valor de la etiqueta depende de la vecindad definida, y como se ha explicado previamente, dicha vecindad puede ser 4-vecinos u 8-vecinos, siendo esta segunda la utilizada de manera predeterminada por *imbinarize*.

Se considerará que el valor del pixel del objeto en una determinada posición es ***I(x,y)=1***, mientas que para una posición del fondo es ***I(x,y)=0***. Además, se tiene en cuenta también que los píxeles que se encuentran fuera de la imagen son 0, o bien, parte del fondo. La región de vecindad es considerada en sentido horizontal primeramente y después en el vertical., tomando como punto de inicio la esquina superior izquierda. En el recorrido, si el pixel actual ***I(x,y)*** es 1, se le asigna una etiqueta nueva, o bien una ya existente en caso de que alguno de sus vecinos ***N(x,y)*** posea una.

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

*Ilustración 23 - Ejemplo de imagen binaria (Fuente: elaboración propia)*

Si dos o más vecinos contienen diferentes etiquetas, se presenta una colisión de etiquetas. Por ejemplo, un objeto con forma de "U" (*Ilustración 23*) tendrá del lado izquierdo y del derecho diferentes etiquetas asignadas, esto es debido a que durante el paso 1, a lo largo de la imagen no era visible que ambos lados estaban conectados por la parte inferior (*ilustración 24*).

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 2 | 2 | 0 | 0 | 3 | 3 | 0 | 0 |
| 0 | 0 | 2 | 2 | 0 | 0 | 3 | 3 | 0 | 0 |
| 0 | 0 | 2 | 2 | 0 | 0 | 3 | 3 | 0 | 0 |
| 0 | 0 | 2 | 2 | 0 | 0 | 3 | 3 | 0 | 0 |
| 0 | 0 | 2 | 2 | 0 | 0 | 3 | 3 | 0 | 0 |
| 0 | 0 | 2 | 2 | 0 | 0 | 3 | 3 | 0 | 0 |
| 0 | 0 | 2 | 2 | 2 | 2 | 2 | 2 | 0 | 0 |
| 0 | 0 | 2 | 2 | 2 | 2 | 2 | 2 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

*Ilustración 24 - Ejemplo de colisión de etiquetas (Fuente: elaboración propia)*

Cuando se presentan dos etiquetas diferentes en un mismo objeto, se dice que se presenta una colisión de etiquetas, que no es tratada directamente en el paso 1, sino que solamente es registrada para su posterior tratamiento en el paso 2. El número de posibles colisiones depende del contenido de la imagen y solo se puede conocer después del paso 1.

El **segundo paso** consiste en resolver las colisiones registradas en el primer paso. Después del recuento de las diferentes etiquetas pertenecientes a un mismo objeto, el valor de la etiqueta de cada pixel es sustituido por una etiqueta general correspondiente a la más pequeña de las contenidas en el.

El proceso de etiquetado en MATLAB también consta de dos pasos. En el primer paso se han de obtener los objetos conectados en una imagen binaria. Para ellos se utiliza la siguiente función:

```
CC = bwconncomp(BW)
```

Dónde **BW** es una imagen binaria obtenida previamente mediante *imbinarize()* y **CC** es una estructura de datos donde se almacenan los resultados.

```
IBRR =

  struct with fields:

    Connectivity: 8
       ImageSize: [480 640]
      NumObjects: 58
    PixelIdxList: {1×58 cell}
```

*Ilustración 25 - Ejemplo de la estructura obtenida de la aplicación de bwconncomp (Fuente Propia)*

Como se puede observar en la ilustración 25, el resultado de utilizar la función *bwconncomp* es una estructura de datos en la que se puede observar:

- **Conectividad aplicada:** por defecto Matlab aplica conectividad-8
- **Tamaño de la imagen:** en el ejemplo la imagen tiene un tamaño de 480x640 píxeles.
- **Número de objetos detectados**
- **PixelIdxList:** cada elemento de la matriz contiene los índices lineales de los píxeles del objeto *k*-ésimo.

El segundo paso es el etiquetado de los objetos detectados mediante *bwconncomp*, para ello existe en MATLAB la siguiente función:

```
L = labelmatrix(CC)
```

Crea una matriz de etiquetas L con los componentes listados en *CC* devueltos por *bwconncomp*

Una vez obtenidas se tienen las imágenes etiquetadas, el último paso para obtener imágenes lo más homogéneas posible es colorear las etiquetas. Para ello MATLAB cuenta con la siguiente función:

```
RGB = label2rgb(L)
```

~~La cual~~ que devuelve imágenes a color según los objetos presentes dentro de la matriz de etiquetas. Por defecto MATLAB aplica el color azul para el fondo

### 3.6.3. Obtención de los valores estadísticos de las imágenes:

Una vez se ha llevado a cabo el proceso de detección de objetos y de etiquetado de los mismos, se procede a calcular el valor medio de la intensidad de todos los píxeles de la imagen obtenida, así como la varianza, dichos valores serán almacenados en una matriz de 3 columnas y una fila por cada imagen. Para ello se utilizan las funciones de MATLAB *med()* y *var()*:

```
media(i,1) = mean(R1,'all');
media(i,2) = mean(G1,'all');
media(i,3) = mean(B1,'all');
```

Los argumentos de entrada de la función *mean* son: R1 es la matriz sobre la cual se va a calcular la media y el parámetro *'all'* le indica a la función que queremos que nos devuelva un único valor en vez de devolver 3 valores, uno por cada capa de color de la imagen de etiquetas coloreadas.

Para poder calcular la varianza, en primer lugar, se ha de proceder a cambiar el tipo de datos que forman las variables R1, G1 y B1:

```
R11 = double(R1);
G11 = double(G1);
B11 = double(B1);
```

El motivo para hacer la operación anterior es que las variables R1, G1 y B1 eran del tipo *uint8*, este tipo de variables consisten en un valor entero de 8 bits positivo, es decir, valores entre 0 y 255. Sin embargo, para poder calcular la varianza es necesario que las variables sean del tipo *single o double*, es decir, números decimales con valores de 32 y 64 bits respectivamente.

Una vez hecho el cambio de tipo de variables se puede proceder a calcular la varianza de las imágenes:

```
varianza(i,1) = var(R11,1,'all');
varianza(i,2) = var(G11,1,'all');
varianza(i,3) = var(B11,1,'all');
```

Los argumentos de entrada de la función *var* son: R11 (la matriz sobre la que se va a calcular la varianza) el parámetro 1 le indica a la función que la varianza va a ser normalizada por todas las observaciones de la muestra, los posibles valores son 0 (por defecto) y 1, para el valor 0 la normalización de la varianza se hace con todas las observaciones de la muestra - 1, y el parámetro *'all'* tiene el mismo efecto que en la función *mean,* es decir, le indica a la función que queremos que nos devuelva un único valor en vez de devolver 3 valores, uno por cada capa de color de la imagen de etiquetas coloreadas.

## 3.7. Bloque 3:

Por último, se procede a comparar los valores de los estadísticos de las imágenes con los obtenidos en los patrones. Para ello este tercer bloque será dividido en 2 etapas:

i. Creación del intervalo de comparación.
ii. Comparación y clasificación.

Sin embargo, en estas dos etapas del tercer bloque los datos de los patrones y de las imágenes se tratan de forma completamente diferente.

### 3.7.1. Creación del intervalo de comparación:

El primer paso para poder comparar y determinar si la imagen de una placa solar es clasificada como una falla es crear un intervalo de comparación. Las imágenes cuyos valores estadísticos permanezcan dentro de dicho intervalo serán consideradas como buenas. Por otro lado, si los valores estadísticos de la imagen quedan fuera del intervalo de comparación, dicha imagen quedará apartada para su posterior revisión.

Las imágenes clasificadas como buenas se las hace corresponder a las placas solares que no tengan ningún tipo de falla. Por otro lado, las placas solares cuyas imágenes queden apartadas tendrán que ser localizadas para ser revisadas.

El intervalo de comparación estará formado por todos los valores que se encuentren entre los valores máximos y mínimos de las medias y las varianzas para cada uno de los canales RGB de los patrones:

$$\left[\min media\,,\max media\right]$$

$$\left[\min varianza\,,\max varianza\right]$$

Para poder almacenar dichos valores, en primer lugar se procede a crear las variables de máximos y mínimos con la función *zeros* que crea un vector de ceros con las dimensiones especificadas por el usuario

```
min_med_patr = zeros(3,1);
max_med_patr = zeros(3,1);
min_var_patr = zeros(3,1);
max_var_patr = zeros(3,1);
```

Una vez se tienen las variables creadas se procede a la búsqueda de los mínimos y lo máximos, utilizando las funciones *min* y *max* de MATLAB:

```
for i = 1:3
    min_med_patr(i) = min(med_patr(:,i));
    max_med_patr(i) = max(med_patr(:,i));
    min_var_patr(i) = min(var_patr(:,i));
    max_var_patr(i) = max(var_patr(:,i));
end
```

### 3.7.2. Comparación y clasificación:

Una vez tenemos creado el intervalo de comparación, solo hay que comparar los valores de las imágenes de prueba con los valores de las imágenes de los patrones. En primer lugar, se crea un vector con valores 0 con la longitud igual al número de imágenes que van a ser comparadas:

```
fallas = zeros(n);
```

Cuando se detecta que una imagen i-ésima se queda fuera del intervalo de comparación, la posición del vector *fallas* equivalente a la imagen i-ésima comparada será cambiado por un número diferente de 0, el número i como se muestra a continuación:

```
for i = 1:n
   for j = 1:3
       if  (med_pru(i,j)  <  min_med_patr(j))  ||  (med_pru(i,j)  >
             max_med_patr(j)) || (var_pru(i,j) < min_var_patr(j)) ||
             (var_pru(i,j) > max_var_patr(j))

          fallas(i)=i;

       end
   end
end
```

Para la clasificación de las imágenes con posibles fallas se parte del vector *fallas* creado, se recorre este vector y, si el valor i-ésimo es diferente de 0, es decir, si almacena la posición de una falla, se carga la imagen del *Datastore* y se guarda en la carpeta fallas con el nombre *"falla_i.jpg"* donde i indica la falla i-ésima encontrada por el algoritmo:

```
for i = 1:n
    if fallas(i) ~= 0
        I = imread(pruebas.Files{i});
        numero = numero + 1;
        formato = '.jpg';
        nombre_completo = carp_fallas + string(numero) + formato;
        imwrite(I,nombre_completo);
    end
end
```

De esta manera todas las imágenes con posibles fallas se almacenarán en la carpeta que este indicada al inicio del programa.

# 4. Resultados:

Los resultados que se exponen a continuación han sido obtenidos mediante la metodología expuesta en el capítulo 3. Desde nuestro punto de vista las pruebas de rendimiento del algoritmo son positivas debido al alto porcentaje de imágenes clasificadas mediante el uso de la media y la varianza de la intensidad de los píxeles de las diferentes imágenes de etiquetas.

Cuando el algoritmo detecte una imagen patrón como falla, el porcentaje de acierto se bajara artificialmente debido a que esa clasificación en concreto es errónea.

## 4.1. Organización de las pruebas de rendimiento:

Tal y como se ha expuesto anteriormente, las imágenes utilizadas para el desarrollo del proyecto han sido agrupadas en 3 tipos diferentes de acuerdo con características de construcción de las placas solares.

Debido a que la resolución de las imágenes, tanto en los patones como en las imágenes de prueba no era uniforme, las pruebas llevadas a cabo se pueden clasificar por resoluciones, y subclasificarlas por tratamiento aplicado a la imagen durante el preprocesado.

Las pruebas agrupadas por resoluciones son las siguientes:

- **Pruebas de igual resolución patrón-experimento:** para este tipo de pruebas ha sido evaluado el porcentaje de imágenes clasificadas correctamente comparando las imágenes de prueba con patrones de la misma resolución.

- **Pruebas con un único tipo de patrones:** para este tipo pruebas ha sido evaluado el porcentaje de imágenes clasificadas correctamente comparando las imágenes de prueba con patrones de un único tamaño, en primer lugar, utilizando los patrones de menor resolución y en segundo lugar se ha realizado la misma metodología, pero con los patrones de mayor resolución.

- **Pruebas con patrones cenitales:** para este tipo pruebas ha sido evaluado el porcentaje de imágenes clasificadas correctamente comparando las imágenes de prueba con patrones de diferentes resoluciones, pero habiendo sido tomadas las imágenes desde un punto de vista cenital, o lo más cenital posible.

Las pruebas por tratamiento son las siguientes:

- **Ninguno:** como el nombre indica las primeras evaluaciones han sido llevadas a cabo sin manipular las diferentes imágenes, ni a los patrones ni las pruebas.

- **Reescalado:** en este segundo tipo de evaluación, se ha procedido a reescalar las imágenes a la resolución de 480x640 píxeles ya que coincide con la resolución de las imágenes con menor resolución.

- **Aumento de nitidez:** en este tercer tipo de evaluación se ha aumentado la nitidez de las imágenes con los parámetros descritos en el capítulo 3.

- **Reescalado y aumento de nitidez:** por último, se ha evaluado el rendimiento del algoritmo realizando un reescalado y un aumento de la nitidez de todas las imágenes.

Cabe destacar que el único tipo sobre el cual se han llevado a cabo todas las pruebas descritas anteriormente ha sido el tipo 1. Esto es debido única y exclusivamente a que los malos resultados obtenidos en las primeras pruebas justifican la ausencia de algunas de las evaluaciones de rendimiento para las imágenes de tipos 2. Para las imágenes de tipo 2 solo se ha evaluado el rendimiento de acuerdo con el criterio de patrones cenitales y, junto a los patrones cenitales, se ha llevado a cabo un reescalado y un aumento de la nitidez. Para las imágenes de tipo 3 no se ha llevado a cabo el reescalado debido a que tanto patrones como imágenes de evaluación son del mismo tamaño, así pues, se ha evaluado el rendimiento sin aplicar ningún tratamiento a las imágenes y aplicando un aumento de la nitidez.

## 4.2. Resumen de las pruebas de rendimiento:

En el anexo B se presentan todos los resultados obtenidos para los tres tipos de imágenes. En este apartado se van a presentar y comentar los resultados obtenidos al aplicar el algoritmo diseñado en este trabajo a los tres tipos de las imágenes obtenidas de las placas solares. Los resultados que se presentan en las tablas 3, 4 y 5 se han seleccionado para mostrar el funcionamiento de este algoritmo de análisis de imágenes. Estos resultados se comparan y analizan como se puede ver en las ilustraciones 28-43 para los tres tipos de imágenes.

### 4.2.1. Pruebas para imágenes de tipo 1:

Únicamente van a ser expuestas en detalle las pruebas presentadas en la siguiente tabla.

| Nº de Prueba | Tratamiento | Tamaños | Nº de Patrones | Nº de Pruebas | Nº de fallas | Nº Clasificadas correctamente | % de acierto |
|---|---|---|---|---|---|---|---|
| 5 | reescalado | todos | 14 | 43 | 11 | 25 | 58,14% |
| 15 | patrones pequeños | todos | 4 | 43 | 30 | 34 | 79,07% |
| 21 | Patrones Cenitales | todos | 8 | 43 | 26 | 34 | 79,07% |
| 27 | Patrones Cenitales nitidez | todos | 8 | 43 | 27 | 35 | 81,40% |

*Tabla 3 -Pruebas que han sido consideradas como favorables para las imágenes de tipo 1*

Estas cuatro pruebas son las que presentan un mejor porcentaje de acierto incluyendo patrones de todas las resoluciones utilizadas en este proyecto y, por lo tanto, permite corroborar que el algoritmo desarrollado tiene un rendimiento bastante aceptable detectando imágenes con posibles fallas.

En primer lugar, la prueba número 5 presenta un porcentaje de éxito bastante bajo. Sin embargo, es la única prueba en la que, teniendo todos los patrones de todos los tamaños, el número de imágenes clasificadas correctamente supera el 50%. A continuación, se muestra la distribución de valores de los estadísticos utilizados:

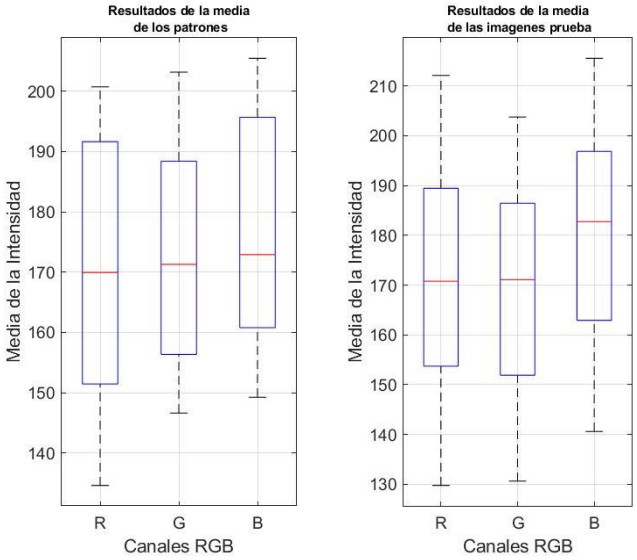


*Ilustración 26 - Prueba 5 - medias (Fuente Propia)*

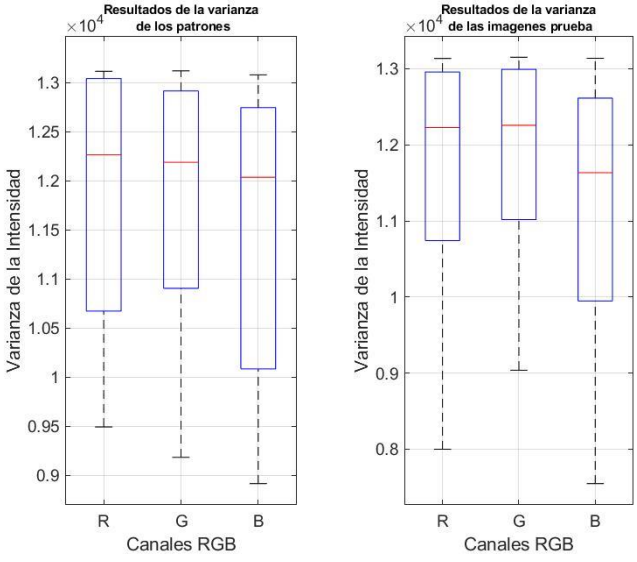


*Ilustración 27 - Prueba 5 - varianzas (Fuente Propia)*

Tal y como se observa en las ilustraciones anteriores, tanto la media como la varianza de los patrones crean un límite muy amplio de valores dentro de los cuales una imagen será considerada como libre de fallas. Por esta razón tan solo el 52% de las imágenes han sido clasificadas correctamente.

En segundo lugar, las pruebas 15 y 21 obtienen el mismo porcentaje de acierto. En la prueba 15 ha sido evaluado el rendimiento utilizando patrones de resolución 480x640 y, para probar el rendimiento, imágenes de todas las resoluciones. Sin embargo, en la prueba 21 han sido utilizados como patrones únicamente aquellas imágenes que han sido tomadas desde un punto de vista cenital o lo más cenital posible, sin tener en cuenta las resoluciones ni de los patrones como ni de las imágenes a analizar. Los resultados de ambas pruebas han sido los siguientes:

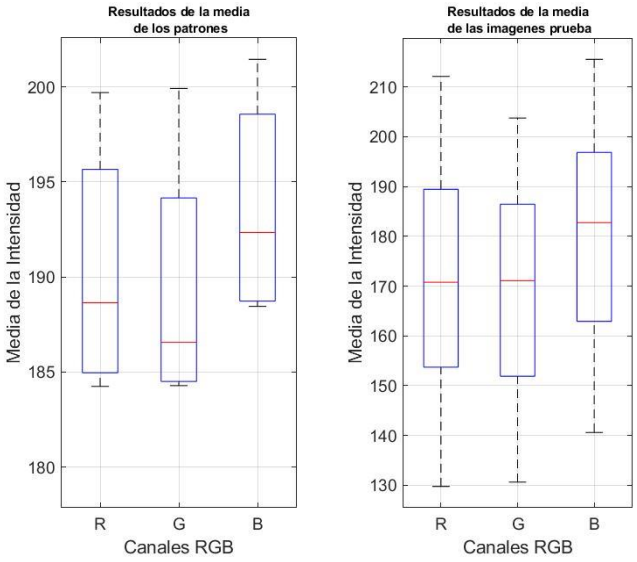


*Ilustración 28 - Prueba 15 - medias (Fuente Propia)*

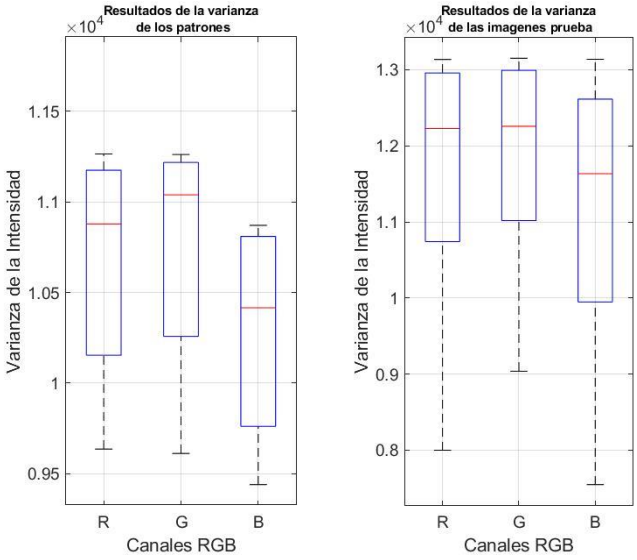


*Ilustración 29 - Prueba 15 - varianzas (Fuente Propia)*

Como se puede observar en las ilustraciones anteriores, los valores mínimos de las medias de los patrones están muy por encima de los valores de las medianas de las imágenes de prueba, facilitando de esta manera el alto porcentaje de éxito. Analizando los valores de las varianzas obtenemos un resultado similar. Sin embargo, en vez de hacer el corte por los valores mínimos, esta vez las varianzas eliminan como imágenes en perfecto estado aquellas que tienen una varianza superior a las de los patrones.

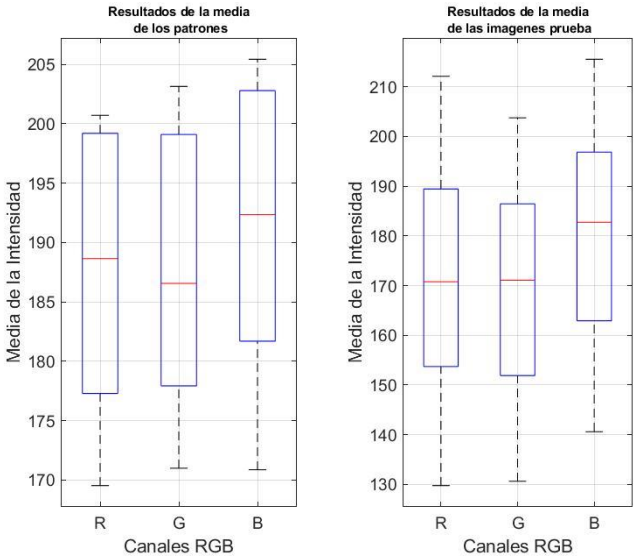


*Ilustración 30 - Prueba 21 - medias (Fuente Propia)*

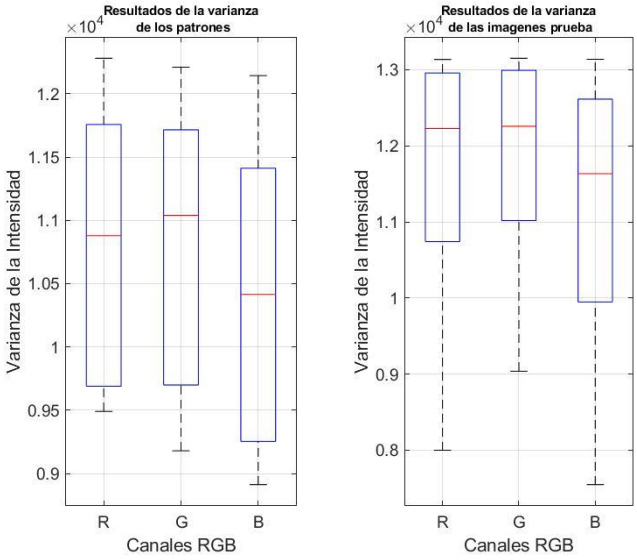


*Ilustración 31 - Prueba 21 - varianzas (Fuente Propia)*

En la prueba 21 se obtiene un resultado similar a la prueba 15, aunque la prueba 21 debe de ser comparada con la prueba 27 debido a que ambas utilizan los mismos patrones, variando el procesado que se les realiza a las imágenes.

Tal y como se ha mostrado en la tabla anterior, las pruebas 21 y 27 han sido llevadas a cabo con los mismos patrones y las mismas imágenes de prueba, con la única diferencia que en la prueba 27 se ha aplicado un aumento de la nitidez de las imágenes con el objetivo de poder captar más detalles, obteniéndose los siguientes resultados:

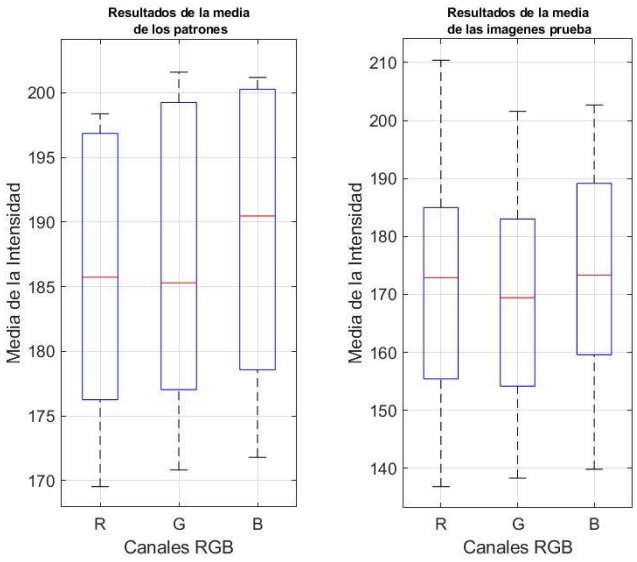


*Ilustración 32 - Prueba 27 - medias (Fuente Propia)*

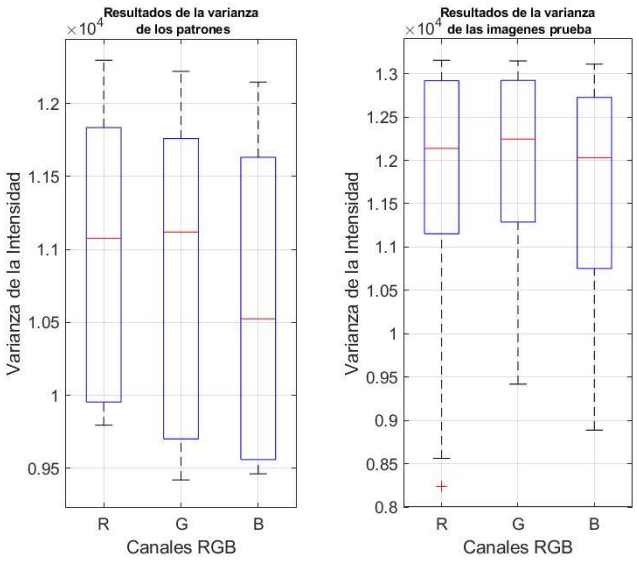


*Ilustración 33 - Prueba 27 - varianzas (Fuente Propia)*

El aumento de nitidez en las imágenes produce que en los patrones los valores mínimos tanto de las medias como de las varianzas aumente, sin variar los valores máximos. Este efecto ocurre también en las imágenes de prueba, pero en menor medida, permitiendo que el algoritmo obtenga un mejor rendimiento y clasifique correctamente una imagen más que si no se aplica un aumento de nitidez.

Observadas todas las pruebas favorables con todos los tipos de imágenes se observa que el algoritmo es capaz de alcanzar un rendimiento bastante elevado, clasificando correctamente el 81% de las imágenes que se le presentan si los patrones con los que se comparan han sido tomados de manera cenital y el rango de resoluciones es el mismo que en el de las imágenes a analizar.

### 4.2.2. Pruebas para imágenes de tipo 2:

Para analizar las imágenes de tipo 2 han sido seleccionadas las dos pruebas siguientes como las más favorables, a pesar de que, como se puede ver en el anexo B, una de las pruebas tuvo un porcentaje de acierto del 100%. Las dos pruebas que se expondrán a continuación tienen porcentajes de acierto superiores al 85% en ambos casos y utilizan imágenes de todas las resoluciones, mientras que la prueba con porcentaje de acierto del 100% utiliza únicamente 2 resoluciones, por ello se ha optado por descartarla a pesar de tener un mejor porcentaje de acierto.

| Nº de Prueba | Tratamiento | Tamaños | Nº de Patrones | Nº de Pruebas | Nº de fallas | Nº Clasificadas correctamente | % de acierto |
|---|---|---|---|---|---|---|---|
| 30 | Patrones Cenitales | todos | 6 | 41 | 30 | 36 | 87,80% |
| 36 | Patrones Cenitales nitidez | todos | 6 | 41 | 29 | 35 | 85,37% |

*Tabla 4 - Pruebas consideradas favorable de las imágenes de tipo 2*

Tal y como se indica en la tabla anterior, como en la tabla completa de pruebas que se puede encontrar en el anexo B, para las imágenes de tipo 2 solo se han llevado a cabo pruebas de rendimiento con patrones cenitales. Esto es debido a que en las pruebas llevadas a cabo con las imágenes de tipo 1, las pruebas con patrones cenitales han sido las que han obtenido mejores resultados con creces, por lo tanto, justifica realizar únicamente pruebas con patrones cenitales para las imágenes de tipo 2, reduciendo así el número de experimentos de 28 para el tipo 1 a 9 para el tipo 2, obteniendo en 8 de ellos porcentajes de acierto superiores al 70%.

A continuación, se presentan los resultados de los estadísticos obtenidos en la prueba 30:

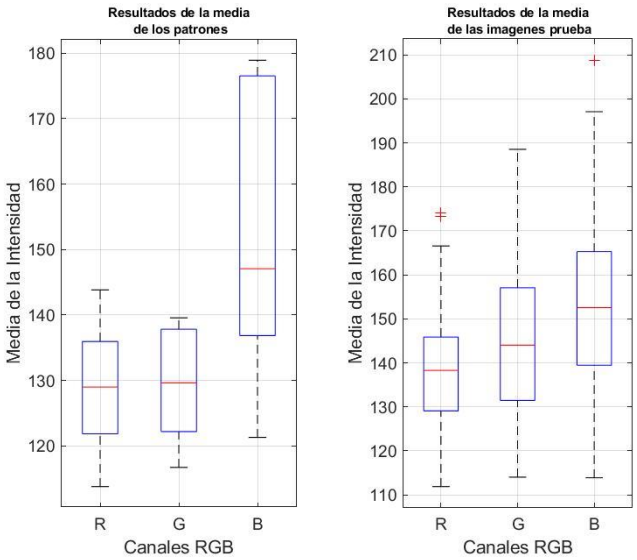


*Ilustración 34 - Prueba 30 - medias (Fuente Propia)*

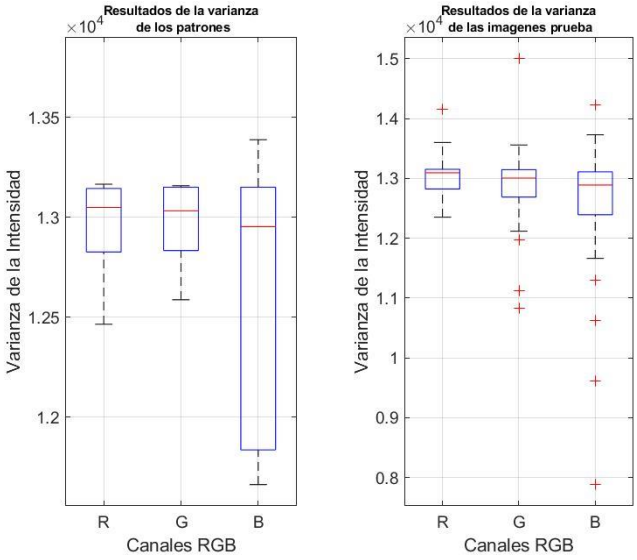


*Ilustración 35 - Prueba 30 - varianzas (Fuente Propia)*

Tal y como se observa en la ilustración de las medias, tanto en los canales rojo como en el verde, los valores máximos de las medias de los patrones esta alrededor de 140 coincidiendo con la mediana de los mismos canales en la batería de imágenes de prueba. A pesar de que en las varianzas esta gran diferencia entre los patrones y las imágenes de prueba no es tan notable, el algoritmo ha conseguido clasificar correctamente el 87,8% de las imágenes.

Entre las pruebas 30 y 36 la única diferencia en el proceso de clasificación es la siguiente: para la prueba 36 se ha llevado a cabo un aumento de la nitidez en todas las imágenes y, a pesar de obtener un resultado más que satisfactorio, ha empeorado la prueba, al contrario que ocurría con las pruebas llevadas a cabo con las imágenes de tipo 1.

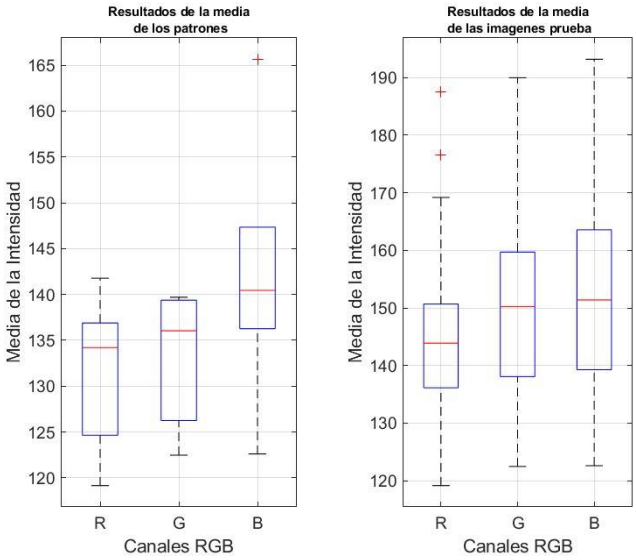


*Ilustración 36 - Prueba 36 - medias (Fuente Propia)*

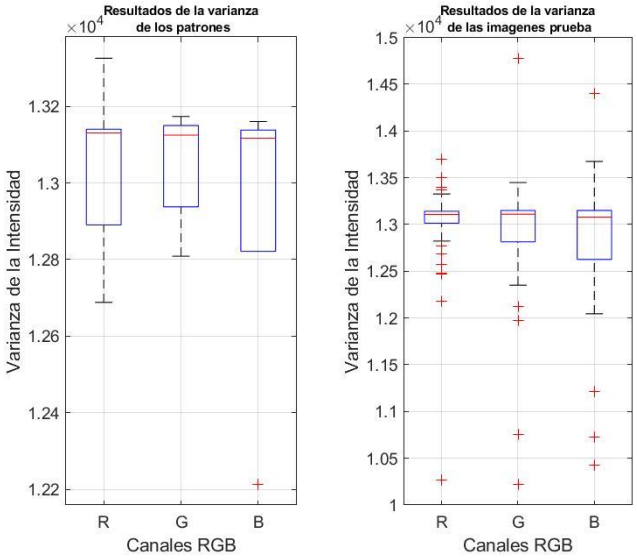


*Ilustración 37 - Prueba 36 - varianzas (Fuente Propia)*

Tal y como sucedía en la prueba 30, en esta prueba el estadístico que ha resultado determinante para la clasificación correcta de las imágenes ha sido la media, superando la mediana de las imágenes de prueba el valor máximo de la media de los patrones en los tres canales de las imágenes. Sin embargo, también se puede observar que en los tres canales existe un conjunto de imágenes cuyo valor de la varianza se encuentra por debajo del mínimo impuesto por los patrones, provocando así la clasificación de dichas imágenes.

### 4.2.3. Pruebas para imágenes de tipo 3:

Para las imágenes del tipo 3, únicamente han sido llevadas a cabo dos pruebas, debido a la falta de imágenes. Los datos que puedan ser recogidos tienen menos relevancia que en las pruebas anteriores ya que, como únicamente hay imágenes de una única resolución no cumplen con el objetivo de flexibilidad planteado en el presente proyecto.

*Tabla 5 - Pruebas consideradas favorable de las imágenes de tipo 3*

| Nº de Prueba | Tratamiento | Tamaños | Nº de Patrones | Nº de Pruebas | Nº de fallas | Nº Clasificadas correctamente | % de acierto |
|---|---|---|---|---|---|---|---|
| 37 | ninguno | 4160x3120 | 4 | 12 | 6 | 10 | 83,33% |
| 38 | nitidez | | 4 | 12 | 5 | 9 | 75,00% |

A pesar de lo expuesto anteriormente los resultados de las últimas pruebas van a ser analizados a continuación:

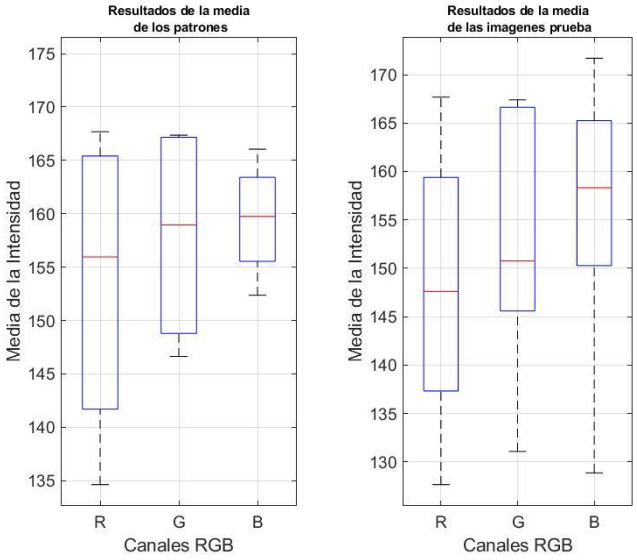


*Ilustración 38 - Prueba 37 - medias (Fuente Propia)*

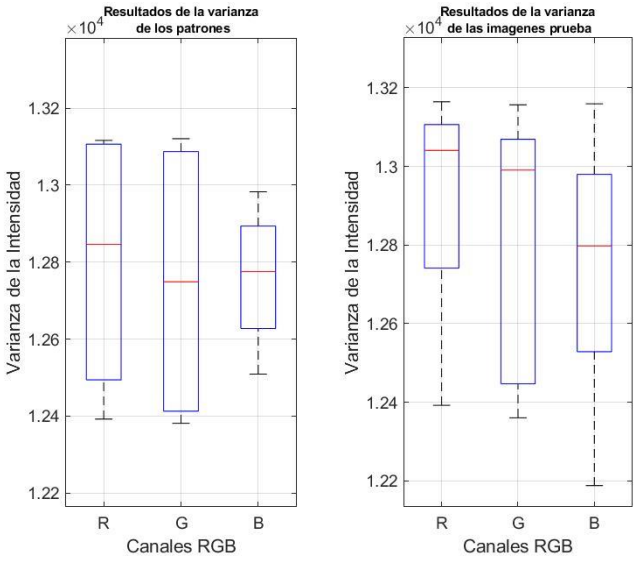


*Ilustración 39 - Prueba 37 - varianzas (Fuente Propia)*

Para la prueba 30, el factor determinante para el alto porcentaje de éxito se encuentra en los valores medios de los canales verde y azul. Una vez más, casi la mitad de los valores de las imágenes de prueba en ambos canales tienen un valor superior a los valores máximos de los respectivos colores en los patrones.

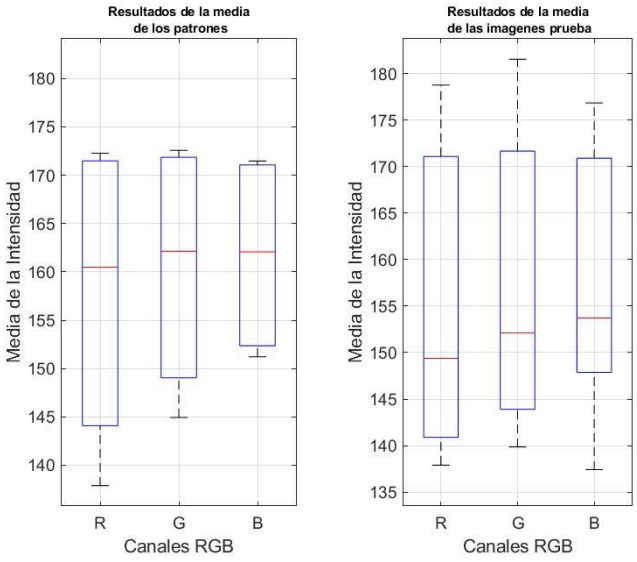


*Ilustración 40 - Prueba 38 - medias (Fuente Propia)*

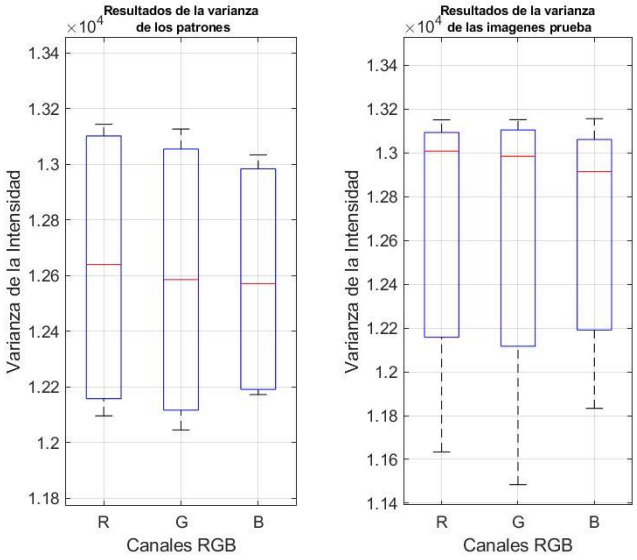


*Ilustración 41 - Prueba 38 - varianzas (Fuente Propia)*

En los resultados de esta última prueba no se observa nada extraordinario ni fuera de lo observado previamente. Existe un cierto número de imágenes por encima de los valores máximos de la media y por debajo del mínimo en las varianzas en los tres canales del espectro RGB.

# Impacto Ambiental:

El análisis del impacto ambiental consiste en la evaluación de las consecuencias ambientales provocadas por el uso de prácticas o tecnologías desarrolladas en un proyecto. Con el objetivo de encontrar estas consecuencias y el grado de agudeza, se realiza un estudio técnico, el objetivo del cual es identificar aquellas prácticas que llevan asociadas las consecuencias, y una vez identificado el problema, tratar de eliminarlo o compensar los impactos negativos derivados del proyecto. Lo que es innegable es que todos los avances o acciones producen algún tipo de impacto ambiental, por lo tanto, mediante el análisis del impacto ambiental se trata de determinar el grado de impacto que se puede considerar aceptable, y reducir el existente hasta conseguir este nivel.

En cuanto al presente proyecto, el conjunto de técnicas y herramientas para el tratamiento digital de imágenes y la posterior clasificación de las mismas si se detecta que pudieran tener algún tipo de falla han sido desarrolladas a través de algoritmos informáticos utilizando un ordenador con el software apropiado (MATLAB) y para la toma de imágenes un teléfono móvil. Por lo tanto el impacto ambiental asociado a este proyecto se reduce a la utilización de la energía eléctrica como fuente de alimentación del ordenador y del teléfono móvil.

Adicionalmente, debido a la necesidad de desplazamiento con el objetivo de obtener las imágenes del conjunto de datos, acudir a reuniones y desplazarse por diferentes bibliotecas o librerías en busca de la bibliografía adecuada, así como para reuniones de control del correcto desarrollo del proyecto, se ha hecho un uso moderado del transporte público del Área Metropolitana de Barcelona, que tiene como impacto ambiental el uso de combustible de los diferentes medios de transporte que forman la red.

# Conclusiones:

Debido al objetivo de flexibilidad que tiene el presente proyecto en lo que resoluciones y dispositivos de toma de imágenes de la primera parte del proyecto únicamente se ha tenido en cuenta el proceso de clasificación manual de los diferentes tipos de imágenes, que van a ser estudiadas siguiendo procesos idénticos de un tipo a otro.

La segunda parte del proyecto consiste en el proceso que se ha llevado a cabo para obtener una serie de imágenes en las que fuera claramente distinguible que forma parte del fondo y que forma parte del objeto de análisis. Para ello se llevó a cabo una separación de la imagen a color en 3 imágenes, cada una para uno de los colores del especto RGB con el fin de obtener la mayor cantidad de información de la imagen original, una vez llevada a cabo esta separación se ha procedido a realizar una segmentación de las imágenes, un etiquetado de las mismas y por último el cálculo de los valores de referencia, en este caso la media y la varianza de la intensidad de las imágenes obtenidas al finalizar el proceso de etiquetado. Durante el proceso de etiquetado se han utilizado diferentes mapas de colores de Matlab a la hora de colorear las etiquetas, siendo elegido el mapa de colores en el cual el fondo queda de color azul por simple preferencia personal, pero cabe decir que el algoritmo funcionaria de igual manera si el fondo fuera de otro color, siempre y cuando el fondo fuera de un color uniforme y todas las imágenes tuvieran el mismo fondo para que pudieran ser considerados útiles los valores obtenidos de dichas imágenes.

Una vez ha sido desarrollado el algoritmo principal ha sido evaluado su funcionamiento variando el tamaño o variando la nitidez de las imágenes con el fin de obtener un tamaño uniforme o destacar los detalles de las imágenes. Una vez evaluados los resultados obtenidos variando estas dos cualidades de las imágenes se han obtenido una serie de resultados bastante favorables llegando a superar el 80% de imágenes clasificadas correctamente, independientemente del equipo con el que han sido tomadas las imágenes respaldando el objetivo de flexibilidad del proyecto.

Hay que destacar el la utilización de dos valores de control es beneficiosa para el funcionamiento del algoritmo, puesto que en las pruebas realizadas se observa que en algunas situaciones la media es el estadístico determinante y en otras la varianza.

Vistos los resultados obtenidos se puede asegurar que el algoritmo tiene una eficacia del 80% si las imágenes patrón han sido tomadas cenitalmente o lo más cenitalmente posible en condiciones de laboratorio con diferentes iluminaciones. Por lo tanto, el siguiente paso que se debería llevar a cabo es el desarrollo de una interfaz gráfica que facilite el uso del programa, así como evaluar el rendimiento del mismo en condiciones reales mediante trabajo de campo. Vistos los resultados obtenidos con el algoritmo una mayor automatización del proceso, así como aumentar el porcentaje de éxito del algoritmo permitiría detectar fallas en grandes instalaciones con la velocidad suficiente como para minimizar las pérdidas de energía que supondría que un panel no funcionase como es debido por suciedad acumulada sobre el mismo o una rotura superficial debida a las condiciones climáticas.

# Presupuesto:

En este apartado se procederá al desglose y justificación del presupuesto. Este consiste en la valoración económica del software utilizado, los costes de ingeniería, materiales y mano de obra. No se ha tenido en cuenta la adquisición de un sistema de toma de las imágenes porque eso depende de cada usuario.

| Costes del proyecto | tarea | horas | Precio/hora | Total |
|---|---|---|---|---|
| Desarrollo del modelo | busqueda de información y estado del arte | 75 | € 40,00 | € 3.000,00 |
| Implementación | Toma de las imágenes | 1,5 | € 40,00 | € 60,00 |
| | Clasificación previa | 1 | € 40,00 | € 40,00 |
| | desarrollo del algoritmo | 215 | € 40,00 | € 8.600,00 |
| Memória del proyecto | metodologias y bibliografia | 30 | € 40,00 | € 1.200,00 |
| | Pruebas de rendimiento | 3 | € 40,00 | € 120,00 |
| | Resultados y conclusiones | 2 | € 40,00 | € 80,00 |
| Estación de trabajo | | | | € 1.100,00 |
| software | licencia de matlab y librerias | | | € 5.000,00 |
| | | | Total | € 19.200,00 |

# Bibliografía:


[1] https://es.wikipedia.org/wiki/Imagen

[2] https://es.wikipedia.org/wiki/Imagen_digital

[3] https://es.wikipedia.org/wiki/Fotograf%C3%ADa_digital

[4] https://es.wikipedia.org/wiki/P%C3%ADxel

[5] https://es.wikipedia.org/wiki/Electr%C3%B3nica_digital

[6] https://es.mathworks.com/help/matlab/ref/image.html

[7] https://es.wikipedia.org/wiki/Imagen_de_mapa_de_bits

[8] https://es.wikipedia.org/wiki/Gr%C3%A1fico_vectorial

[9] E. Cuevas, M. Díaz Cortés, J.O. Camarena Méndez, "Tratamiento digital de imágenes con MATLAB", Marcombo 2018.

[10] A. Gilat, "MATLAB Una introducción con ejemplos práctico" Editorial Reverte 2006

[11] https://es.mathworks.com/help/images/ref/imbinarize.html?s_tid=doc_ta

[12] https://es.mathworks.com/help/images/ref/bwconncomp.html?s_tid=doc_ta

[13] https://es.mathworks.com/help/images/ref/labelmatrix.html?s_tid=doc_ta

[14] https://es.mathworks.com/help/images/ref/label2rgb.html?s_tid=doc_ta

[15] https://es.mathworks.com/help/matlab/ref/matlab.io.datastore.imagedatastore.html?s_tid=doc_ta

[16] https://es.mathworks.com/help/matlab/ref/uint8.html?s_tid=doc_ta

[17] https://es.mathworks.com/help/matlab/ref/mean.html?s_tid=doc_ta

[18] https://es.mathworks.com/help/matlab/ref/var.html?s_tid=doc_ta

[19] https://es.mathworks.com/help/releases/R2018b/images/ref/locallapfilt.html

[20] https://es.wikipedia.org/wiki/Panel_solar

[21] https://es.wikipedia.org/wiki/Energ%C3%ADa_solar

# Anexo A:

## A.1. Código de Matlab del proyecto:

### A.1.1. Programa Principal:

```
%Limpieza del workspace
close all
clear all
clc

%Dirección de las carpetas donde están las imágenes
carp_patr = 'fotos/tipo 3/patrones';
carp_pru = 'fotos/tipo 3/';
carp_fallas = 'fotos/tipo 3/fallas/falla_';

%variable para nombrar las tablas de excel donde se recogen los
datos
%únicamente se utiliza en las pruebas de rendimiento o si se
pretende
%exportar los datos
num_exp = 38;

%cargamos todas las imágenes
patrones = imageDatastore(carp_patr);
pruebas = imageDatastore(carp_pru);


%estraccion de los estadísticos de los patrones
[med_patr,var_patr,m] = estadisticos(patrones);
min_med_patr = zeros(3,1);
max_med_patr = zeros(3,1);
min_var_patr = zeros(3,1);
max_var_patr = zeros(3,1);

for i = 1:3
    min_med_patr(i) = min(med_patr(:,i));
    max_med_patr(i) = max(med_patr(:,i));
    min_var_patr(i) = min(var_patr(:,i));
    max_var_patr(i) = max(var_patr(:,i));
end

%Calculamos los estadísticos de las images con posibles fallas
[med_pru,var_pru,n] = estadisticos(pruebas);

fallas = zeros(n);
%Comparación de los estadísticos patrón con los datos de prueba
for i = 1:n
    for j = 1:3
        if (med_pru(i,j) < min_med_patr(j)) || (med_pru(i,j) >
max_med_patr(j)) || (var_pru(i,j) < min_var_patr(j)) ||
(var_pru(i,j) > max_var_patr(j))
            fallas(i)=i;
        end
    end
end
```

```
%separación de las imagenes en una carpeta de fallas
numero = 0;
for i = 1:n
    if fallas(i) ~= 0
        I = imread(pruebas.Files{i});
        numero = numero + 1;
        formato = '.jpg';
        nombre_completo = carp_fallas + string(numero) +
formato;
        imwrite(I,nombre_completo);
    end
end

%%
%Escritura de los datos en un archivo excel para su posterior
analisis
nom_med_patr = 'resultados/Prueba_';
nom_comp_med_patr = nom_med_patr + string(num_exp);

xlswrite(nom_comp_med_patr,med_patr,1);
xlswrite(nom_comp_med_patr,var_patr,2);
xlswrite(nom_comp_med_patr,med_pru,3);
xlswrite(nom_comp_med_patr,var_pru,4);

%impresion por linea de comandos las imágenes con fallas
detectadas
fprintf('El número de imágenes con fallas detectadas es: %u\n',
numero)
```

## A.1.2. Función estadísticos:

```
function [media,varianza,n] = estadisticos(imagenes)

    %obtenemos el número total de imágenes
    n = length(imagenes.Files);

    %creamos las variables que almacenarán los valores de las
medias y las
    %varianzas de las imágenes patrón
    media = zeros(n,3);
    varianza = zeros(n,3);

    for i = 1:n
        %seleccionamos una imágen en concreto
        I = imread(imagenes.Files{i});

        %Los dos comandos siguientes solo se utilizarán para
las pruebas
        %pertinentes
        %reescalado de las imágenes
        I = imresize(I,[480,640]);
        %aumento de la nitidez de la imagen
        I = locallapfilt(I,1,0.65);


        %separar la imagen en 3 imagenes, una para cada color
        [I1RG,I1GG,I1BG] = separar_colores(I);

        %Obtención de las imágenes binárias
        IBR = imbinarize(I1RG);
        IBG = imbinarize(I1GG);
        IBB = imbinarize(I1BG);

        %Obtención del número de objetos en las diferentes
imágenes
        IBRR = bwconncomp(IBR);
        IBGG = bwconncomp(IBG);
        IBBB = bwconncomp(IBB);
        %Etiquetado de las imágenes con los objetos detectados
        LR = labelmatrix(IBRR);
        LG = labelmatrix(IBGG);
        LB = labelmatrix(IBBB);

        %Coloreado de las etiquetas
        R1 = label2rgb(LR);
        G1 = label2rgb(LG);
        B1 = label2rgb(LB);

        %Cálculo de los dos estadísticos utilizados
        %media
        media(i,1) = mean(R1,'all');
        media(i,2) = mean(G1,'all');
        media(i,3) = mean(B1,'all');

        %Pasamos las imágenes a double, la función var() no
admite valore
```

```
            %uint8
            R11 = double(R1);
            G11 = double(G1);
            B11 = double(B1);

            %Varianza
            varianza(i,1) = var(R11,1,'all');
            varianza(i,2) = var(G11,1,'all');
            varianza(i,3) = var(B11,1,'all');

        end
    end
```

### A.1.3. Función separar colores:

```
function [I1R, I1G, I1B] = separar_colores(I)
%Separa la imagen I1 proporcionada en 3 imágenes, cada una de
las cuales
%Guarda los datos de cada canal RGB

    %En primer lugar se obtienen las dimensiones de la imagen
    IB = rgb2gray(I);

    %[anchoI, altoI] = size(IB);
    I1 = im2double(I);

    %En segundo lugar se guardan los valores de RGB en cada una
de las nuevas
    %imágenes
    %Imágen resultado del canal R
    I1R = I1(:,:,1);
    %Imágen resultado del canal G
    I1G = I1(:,:,2);
    %Imágen resultado del canal B
    I1B = I1(:,:,3);

end
```

## A.1.4. Fichero de Prueba de nitidez:

```
%Algoritmo para probar diferentes configuraciones de parametros
para el aumento de la nitidez

%Limpieza del workspace
close all
clear all
clc

%Dirección de las carpetas donde están las imágenes
carpeta ='nitidez';
%num_exp = 8;

%cargamos todas las imágenes
pruebas = imageDatastore(carpeta);
n = length(pruebas.Files);

for i = 1:n
    I = imread(pruebas.Files{i});
    I1 = locallapfilt(I,1,0.65);

    figure()
    subplot(1,2,1);
    imshow(I);
    subplot(1,2,2);
    imshow(I1);
end
```

## A.2. Código de Matlab de las gráficas de resultados:

```
close all
clear all
clc
%%
% n es el numero total de experimentos, se determina una vez se
han
% finalizado todas las pruebas
n = 12;

for i = 1:n
    nom_xls = 'resultados/Prueba_' + string(i);

    med_patr = xlsread(nom_xls,1);
    var_patr = xlsread(nom_xls,2);
    med_pru = xlsread(nom_xls,3);
    var_pru = xlsread(nom_xls,4);

    [m_med_patr, n] = size(med_patr);
    [m_var_patr, n2] = size(var_patr);
    [m_med_pru, n3] = size(med_pru);
    [m_var_pru, n4] = size(var_pru);

    canales = {'R','G','B'};

    Imed = figure('Name','Resultado de la Prueba 1');
    subplot(1,2,1);
    boxplot(med_patr,canales);
    ylabel('Media de la Intensidad');
    xlabel('Canales RGB');
    title('Resultados de la media de los
patrones','FontSize',8);
    grid on;
    subplot(1,2,2);
    boxplot(med_pru,canales);
    ylabel('Media de la Intensidad');
    xlabel('Canales RGB');
    title('Resultados de la media de las imagenes
prueba','FontSize',8);
    grid on;

    nombre_med = 'resutlados/Prueba_' + string(i) + 'med';

    print(Imed,nombre_med,'-djpeg');

    Ivar = figure('Name','Resultado de la Prueba 1');
    subplot(1,2,1);
    boxplot(var_patr,canales);
    ylabel('Varianza de la Intensidad');
```

```
        xlabel('Canales RGB');
        title('Resultados de la media de los patrones');
        grid on;
        subplot(1,2,2);
        boxplot(var_pru,canales);
        ylabel('Varianza de la Intensidad');
        xlabel('Canales RGB');
        title('Resultados de la media de las imagenes prueba');
        grid on;

        nombre_var = 'resutlados/Prueba_' + string(i) + 'var';

        print(Imed,nombre_var,'-djpeg');
    end
```

# Anexo B:

## B.1. Tabla de resultados completa:

*Tabla 6 - Resultados de las pruebas con imágenes de tipo 1*

| Nº de Prueba | Tratamiento | Tamaños | Nº de Patrones | Nº de Pruebas | Nº de fallas | Nº Clasificadas correctamente | % de acierto |
|---|---|---|---|---|---|---|---|
| 1 | ninguno | todos | 14 | 43 | 11 | 25 | 58,14% |
| 2 | ninguno | 480x640 | 4 | 8 | 4 | 8 | 100,00% |
| 3 | ninguno | 480x640 y 1200x1600 | 4 | 22 | 15 | 19 | 86,36% |
| 4 | ninguno | 4160x3120 | 10 | 21 | 4 | 14 | 66,67% |
| 5 | reescalado | todos | 14 | 43 | 11 | 25 | 58,14% |
| 6 | reescalado | 480x640 y 1200x1600 | 4 | 22 | 15 | 19 | 86,36% |
| 7 | reescalado | 4160x3120 | 4 | 21 | 4 | 8 | 38,10% |
| 8 | nitidez | todos | 14 | 43 | 7 | 21 | 48,84% |
| 9 | nitidez | 480x640 | 4 | 8 | 4 | 8 | 100,00% |
| 10 | nitidez | 480x640 y 1200x1600 | 4 | 22 | 15 | 19 | 86,36% |
| 11 | nitidez | 4160x3120 | 10 | 21 | 3 | 13 | 61,90% |
| 12 | reescalado y nitidez | todos | 14 | 43 | 7 | 21 | 48,84% |
| 13 | patrones pequeños | 480x640 y 1200x1600 | 4 | 22 | 15 | 19 | 86,36% |
| 14 | patrones pequeños | 4160x3120 | 4 | 21 | 20 | 20 | 95,24% |
| 15 | patrones pequeños | todos | 4 | 43 | 30 | 34 | 79,07% |
| 16 | patrones grandes | 480x640 y 1200x1600 | 10 | 22 | 6 | 16 | 72,73% |
| 17 | patrones grandes | 4160x3120 | 10 | 21 | 4 | 14 | 66,67% |
| 18 | patrones grandes | todos | 10 | 43 | 11 | 21 | 48,84% |
| 19 | Patrones Cenitales | 480x640 y 1200x1600 | 8 | 22 | 12 | 16 | 72,73% |
| 20 | | 4160x3120 | 8 | 21 | 13 | 17 | 80,95% |
| 21 | | todos | 8 | 43 | 26 | 34 | 79,07% |
| 22 | Patrones Cenitales con reescalado | 480x640 y 1200x1600 | 8 | 22 | 12 | 20 | 90,91% |
| 23 | | 4160x3120 | 8 | 21 | 13 | 17 | 80,95% |
| 24 | | todos | 8 | 43 | 26 | 34 | 79,07% |
| 25 | Patrones Cenitales nitidez | 480x640 y 1200x1600 | 8 | 22 | 14 | 18 | 81,82% |
| 26 | | 4160x3120 | 8 | 21 | 12 | 17 | 80,95% |
| 27 | | todos | 8 | 43 | 27 | 35 | 81,40% |

*Tabla 7 - Resultados de las pruebas con imágenes de tipo 2*

| Nº de Prueba | Tratamiento | Tamaños | Nº de Patrones | Nº de Pruebas | Nº de fallas | Nº Clasificadas correctamente | % de acierto |
|---|---|---|---|---|---|---|---|
| 28 | Patrones Cenitales | 480x640 y 1200x1600 | 6 | 21 | 16 | 18 | 85,71% |
| 29 | | 4160x3120 | 6 | 21 | 12 | 16 | 76,19% |
| 30 | | todos | 6 | 41 | 30 | 36 | 87,80% |
| 31 | Patrones Cenitales reescalado | 480x640 y 1200x1600 | 6 | 21 | 18 | 20 | 95,24% |
| 32 | | 4160x3120 | 6 | 21 | 9 | 11 | 52,38% |
| 33 | | todos | 6 | 41 | 26 | 32 | 78,05% |
| 34 | Patrones Cenitales nitidez | 480x640 y 1200x1600 | 6 | 21 | 19 | 21 | 100,00% |
| 35 | | 4160x3120 | 6 | 21 | 11 | 15 | 71,43% |
| 36 | | todos | 6 | 41 | 29 | 35 | 85,37% |

*Tabla 8 - Resultados de las pruebas con imágenes de tipo 3*

| Nº de Prueba | Tratamiento | Tamaños | Nº de Patrones | Nº de Pruebas | Nº de fallas | Nº Clasificadas correctamente | % de acierto |
|---|---|---|---|---|---|---|---|
| 37 | ninguno | 4160x3120 | 4 | 12 | 6 | 10 | 83,33% |
| 38 | nitidez | | 4 | 12 | 5 | 9 | 75,00% |

# Anexo C:

## C.1. Lista de tablas:



## C.2. Lista de ilustraciones: