

Grado Universitario en Ingeniería electrónica, industrial y
automática.
2017-2018

Trabajo Fin de Grado

“Detección automática de defectos en artículos de cerámica”

Ángela Muñoz Moreno-Arrones

Tutor:

Luis Santiago Garrido Bullón

Leganés, Julio 2018.



[Incluir en el caso del interés de su publicación en el archivo abierto]

Esta obra se encuentra sujeta a la licencia Creative Commons
Reconocimiento – No Comercial – Sin Obra Derivada



RESUMEN

El proyecto “Detección automática de defectos en artículos de cerámica” surge de la adaptación de las fábricas industriales a los nuevos métodos de detección y extracción de características asociados a la visión por computador, que permiten de forma más eficiente y menos costosa la identificación, en este caso, de defectos en platos.

Por un lado, a partir de la visión por computador se explicarán las etapas que lo componen y los factores que participan en cada una de ellas, así como los posibles métodos a aplicar como las transformaciones morfológicas.

Por otro lado, se tratarán aspectos relacionados con un entorno industrial controlado, la planificación del proyecto y el impacto socioeconómico de automatizar el proceso de detección de defectos en la industria.

Palabras clave: visión por computador, transformaciones morfológicas, industrial, identificación, impacto socioeconómico.

RESUME

The project “Automatic detection of defects in ceramic products” comes out of the adaptation of the industrial manufacturers to the new methods of detection and extraction of computer vision related characteristics that allow in this case low-cost and more efficient detection of defects in plates.

On the one hand, we will start from defining computer vision and the different phases that are part of it, as well as the factors that take part in every step of the process and the best methods to apply in each phase such as morphological transformation.

On the other hand, we will talk about a controlled industrial environment, the planification of the project and the socioeconomic impact of automating the defect detection process in the industry.

Key words: computer vision, morphological transformation, industrial, identification, socioeconomic impact.

DETECCIÓN AUTOMÁTICA DE DEFECTOS EN ARTÍCULOS DE CERÁMICA





DEDICATORIA



CONTENIDO

1. INTRODUCCIÓN	7
1.1 MOTIVACIÓN	7
1.2 OBJETIVOS.....	7
2. ESTADO DEL ARTE	8
2.1 PROCESAMIENTO DE IMÁGENES	9
2.1.1 ADQUISICIÓN DE LA IMAGEN	9
2.1.2 FILTRADO.....	10
2.1.3 SEGMENTACIÓN	10
2.2 FUZZY	10
2.2 LAPLACIAN OF GAUSSIAN FILTER	12
2.3 GRADIENTE.....	12
2.4 CANNY	14
3. PRUEBAS	16
3.1 ELECCIÓN DEL FILTRO	23
3.1.1 GRADIENTE.....	24
3.1.2 CANNY	28
3.1.3 LAPLACIAN OF GAUSSIAN FILTER	28
4. DISEÑO E IMPLEMENTACIÓN	30
4.1 TRANSFORMACIONES MORFOLÓGICAS.....	30
4.2 RESTA DE IMÁGENES	30
4.3 ELIMINACIÓN DE RUIDO	30
4.4 ANÁLISIS DEL DEFECTO	30
5. RESULTADOS	35
5.1 ROTURA PARCIAL.....	35
5.2 ROTURA EN EL BORDE	40
5.3 DESCASCARILLAMIENTO DEL BORDE	45
5.4 GOLPE PARTE TRASERA	50
5.5 PLATO SIN DEFECTOS.....	55
6. CONCLUSIONES	60
7. TRABAJO FUTURO	61
8. ANEXO.....	62
8.1 ASPECTO SOCIOECONÓMICO	62
8.2 MARCO REGULADOR	63
8.3 PLANIFICACIÓN DEL PROYECTO	65
8.4 PRESUPUESTO	66
8.5 CÓDIGO	67
8.5.1 PROGRAMA	67
8.5.2 FUNCIÓN OBTENER DEFECTO	76
9. ÍNDICE DE FIGURAS.....	78
10. ÍNDICE DE TABLAS	80
11. BIBLIOGRAFÍA	81



1. INTRODUCCIÓN

1.1 MOTIVACIÓN

El uso de la robótica en el entorno industrial ha aumentado durante los últimos años permitiendo así una mayor productividad en las fábricas y reduciendo los tiempos de fabricación. El impacto económico en las ciudades que cuentan con industrias con un elevado grado de automatización ha permitido el desarrollo de las mismas.

En cuanto a la detección de defectos en los artículos producidos en las plantas industriales, tradicionalmente se encontraban varios trabajadores identificando uno a uno los defectos de dichos productos, pero este método no era eficiente debido entre otros a que el trabajador no siempre aplicaba el mismo criterio para desechar el artículo según el defecto presente. La detección automática de defectos permite unificar las decisiones tomadas respecto a desechar o no los platos si éstos presentan un defecto que no permita que el plato sea enviado a la tienda y comprado por el cliente.

1.2 OBJETIVOS

Este proyecto se basa en la creación de un programa realizado en Matlab para poder detectar defectos en artículos de cerámica en tiempo real mediante la implementación del código en una fábrica en Rumania dedicada a la fabricación de platos, la cual mantiene un contrato exclusivo con IKEA.

El objetivo principal es que se aumente la productividad en la fábrica al eliminar el cuello de botella provocado por esta etapa de detección y poder reducir así el número de errores producidos en esta fase. El código por implementar no debe permitir que los platos con defectos sean enviados a las tiendas, sino que se destruyan los mismos si el defecto presente en ellos presenta unas características que no permiten que el plato en cuestión cumpla con los estándares de calidad marcados por la marca y por la normativa europea, ya que hay que tener en cuenta que es un objeto que se va a emplear en acciones relacionadas con la alimentación.

En un apartado posterior de la memoria expondremos el [marco regulador](#) en cuanto a estándares técnicos y el análisis de la legislación aplicable sobre la implantación del método desarrollado en este proyecto.



2. ESTADO DEL ARTE

Para poder definir el procesamiento digital de las imágenes, tendremos que conocer en primer lugar qué es una imagen.

Una imagen se puede definir como una función bidimensional $f(x,y)$ cuyo espacio de entrada se denomina intensidad de la imagen para un punto en concreto de la misma. Nos referimos a una imagen digital, cuando los valores de intensidad de nuestra función bidimensional, así como las coordenadas correspondientes a la misma son finitas. Todas las imágenes están compuestas por puntos o píxeles, los cuales toman valores enteros que, en las imágenes en color para algunos formatos, de 24 bits, estarán comprendidos entre 0 y 16.777.216. La visión por computador corresponde a una sección de la inteligencia artificial que se encarga de replicar la inteligencia humana y que hace uso de las imágenes. Podemos diferenciar varios niveles que separan la visión por computador del procesamiento de imágenes. En primer lugar, nos podremos quedar en el primer nivel entre ambas partes de tal forma que únicamente realicemos la fase de preprocesamiento, de tal forma que eliminemos aquellas características que nos impidan analizar la imagen o realcemos otras que sean importantes. El segundo nivel consiste en transformar las imágenes que resultan atractivas para el ojo humano en otras más simples que puedan ser fácilmente analizables por un computador, de tal forma que obtengamos a la salida del proceso las características detectadas. Finalmente, encontramos aquellas tareas que están relacionadas con dar un sentido a la imagen tras analizar los elementos de la misma de forma individual, lo cual está relacionado con la función cognitiva de la visión.

Para explicar de forma gráfica la visión por computador, pondremos un ejemplo real de aplicación de la misma. En las piscinas se utiliza visión por computador como apoyo a los socorristas para evitar que los nadadores se ahoguen dentro del agua. Se trata de un sistema diseñado mediante cámaras que en tiempo real identifica a las personas y también su movimiento, de tal forma que cuando un individuo se encuentra parado un tiempo superior a cinco segundos en el fondo de la piscina se envía una señal sonora al socorrista para que acuda a rescatarle. ¿Cómo identificamos en este ejemplo los tres escalones definidos anteriormente? detectaremos las características físicas de las personas de tal forma que se consiga realizar un seguimiento de las mismas (altura, constitución...) y daremos un sentido a todos los valores registrados para finalmente detectar el ahogamiento y realizar el aviso pertinente.

La visión permite al ser humano obtener información de la posición de los objetos del entorno, análisis del color o la presencia de posibles peligros. A finales de la década de los sesenta se impulsó el desarrollo de la visión por computador con los primeros avances en los computadores, con la convicción de conseguir de forma rápida estructuras tridimensionales de imágenes para a partir de ellas entender el conjunto de la misma. El problema inicial de la visión por computador era que se quería imitar la forma de ver y procesar las imágenes por el ojo humano, pero no se sabía exactamente cómo se producía esta transmisión de información, lo cual resultaba un problema. La investigación en este campo aumentó de forma progresiva a la par que disminuía el coste y aumentaba la eficiencia de los nuevos ordenadores y hardware. La visión por computador permite obtener una elevada cantidad de información de imágenes digitales o videos, de tal forma que se basa en un sistema que busca mejorar o asemejarse a la detección realizada por el ser humano. Lo que buscamos es extraer de forma automática y analizar información relevante de una imagen o una secuencia de imágenes y este método es aplicable a numerosos ámbitos, siendo la industria uno de ellos.

La visión computador se puede emplear para buscar un elemento concreto dentro de una imagen, por ejemplo, diferenciar entre varias personas. Una aplicación directa de la visión por computador que se utiliza hoy en día consiste en la identificación de la matrícula de los automóviles que se saltan un semáforo o el número de vehículos que atraviesa un puente en hora punta. A continuación, se puede observar un ejemplo de este método:

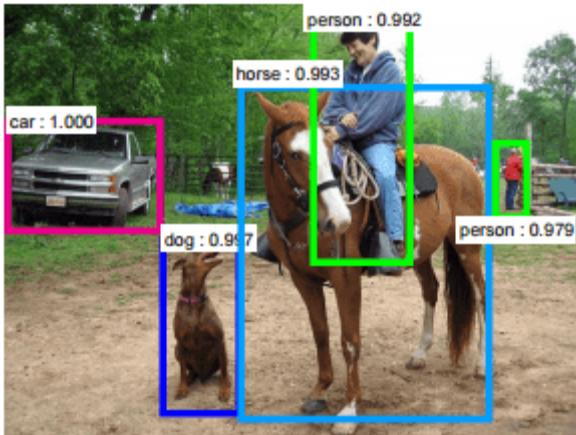


Fig. 1: Ejemplo visión por computador.

En la figura anterior, podemos observar una aplicación de la visión por computador. Los objetos que han sido identificados en la imagen se recuadran y se hace referencia a la etiqueta correspondiente, mostrando así el porcentaje de que dicho objeto pertenezca a la clase identificada. Otra de las aplicaciones de la visión por computador es la detección de enfermedades cancerígenas.

Este método ha permitido un gran avance en las técnicas de identificación y ha superado de manera notable las expectativas iniciales.

A continuación, identificaremos las diversas fases que componen la visión por computador.

2.1 PROCESAMIENTO DE IMÁGENES

El proceso que transcurre desde que tomamos una imagen en la fábrica hasta que obtenemos la información útil de ella está compuesto por varias etapas que nos van a permitir procesar la imagen y realizar sobre ella las transformaciones necesarias para que sea fácilmente analizable por un computador y realzar aquellas características que nos interesan y eliminar aquellas que nos impiden analizar la imagen correctamente.

2.1.1 ADQUISICIÓN DE LA IMAGEN

El primer paso será el de tomar o adquirir la imagen, lo cual se puede realizar mediante una cámara fija. En nuestro caso, la fábrica contará con un entorno controlado compuesto por una cámara fija y una iluminación constante para poder evitar obtener errores debido a estos factores y aumentar la probabilidad de éxito del método, es decir, suponemos que la iluminación es constante entre dos imágenes sucesivas y que la cámara permanecerá fija en ambas.



2.1.2 FILTRADO

Para poder analizar correctamente las imágenes, necesitaremos eliminar aquellas características que nos impiden dicho propósito y/o potenciar o favorecer las que queremos estudiar. Dentro de la etapa de preprocesamiento tendremos en cuenta la eliminación de ruido.

2.1.3 SEGMENTACIÓN

La segmentación de la imagen digital está basada en tres premisas, por un lado, la conectividad, es decir, que los píxeles de un mismo objeto se encuentran agrupados, por otro lado, la discontinuidad, los objetos tienen unos bordes bien definidos que permiten que éstos resalten del fondo y sean más fácilmente identificables, y por último la similitud, los píxeles de un mismo objeto presentan características similares en cuanto a textura y color. La segmentación de nuestro objeto la haremos en base a la umbralización, es decir, trabajaremos directamente con el histograma e identificaremos los píxeles pertenecientes al objeto en blanco y en negro a los del fondo. Utilizaremos el algoritmo de la umbralización debido a que los píxeles del objeto presentan unos niveles de gris muy diferentes a los del fondo.

Lo que buscamos por tanto con este proyecto es aumentar la eficacia y reducir el coste de la detección de defectos en artículos de cerámica en una fábrica industrial, ya que dicha fábrica se encuentra completamente automatizada como explicaremos más adelante, y uno de los cuellos de botella que presenta es la fase de detección de defectos de los artículos de cerámica que van siendo transportados por la cinta.

A continuación, expondremos los diferentes métodos de detección de bordes, concepto en el que se basa la detección de defectos.

2.2 FUZZY

La teoría de fuzzy fue por primera vez introducida hace más de cuarenta años y permite trabajar con información imprecisa. La lógica difusa es una forma de asignar un espacio de entrada a un espacio de salida, de tal forma que no es un tipo de lógica basada en la precisión sino en una respuesta aproximada y en el lenguaje natural. Mediante la lógica difusa podemos modelar funciones no lineales amparándonos en conceptos matemáticos muy simples. Podemos definir por tanto a la lógica difusa como un modelo intuitivo que podemos incluir en nuestro sistema al computar las palabras o estados que definimos en él.

Definimos Z como un conjunto de elementos u objetos con un elemento genérico z de tal forma que $Z = \{ z \}$. Cada elemento tiene asociado un número real entre 0 y 1, de tal forma que un conjunto de fuzzy A en Z está caracterizado por una función de pertenencia. La diferencia con los conjuntos ordinarios es que éstos sólo toman valores binarios (0 ó 1) dependiendo de si pertenecen o no los elementos al conjunto, mientras que en este caso los elementos tomarán los valores extremos 0 ó 1 con la misma interpretación anterior, pero también podrán tener valores intermedios indicando que pertenecen parcialmente a un conjunto.

De esta forma, identificamos un conjunto fuzzy como un conjunto ordenado de valores de z y su correspondiente función de pertenencia que establece el grado de pertenencia a cada z , con lo que obtenemos la ecuación 1.

$$A = \{z, \mu_A(z) \mid z \in Z\} \quad (1)$$

Cuando las variables que tratamos son continuas el conjunto A en la ecuación (1) puede tomar una cantidad infinita de elementos, mientras que cuando los valores de z son discretos podemos mostrar los elementos de A de forma explícita.

A partir de los conjuntos caracterizados mediante fuzzy podemos definir varias propiedades tales como:

1. El conjunto vacío o conjunto de elementos cuya función de pertenencia en Z es igual a 0.
2. Igualdad. Dos conjuntos de elementos A y B son iguales sólo si $\mu_A(z) = \mu_B(z)$ para todo $z \in Z$.
3. El complementario de un conjunto fuzzy se define como el conjunto cuya función de pertenencia es:

$$\mu_{\bar{A}}(z) = 1 - \mu_A(z) \quad \forall z \in Z \quad (2)$$

4. El subconjunto de un conjunto fuzzy A es el subconjunto B sólo si:

$$\mu_A(z) \leq \mu_B(z) \quad \forall z \in Z \quad (3)$$

5. La unión (OR) de dos conjuntos de fuzzy A y B da como resultado un conjunto C con la siguiente función de pertenencia:

$$\mu_U(z) = \max [\mu_A(z), \mu_B(z)] \quad \forall z \in Z \quad (4)$$

6. La intersección (AND) de dos conjuntos de fuzzy A y B da como resultado un conjunto I con la siguiente función de pertenencia:

$$\mu_I(z) = \min[\mu_A(z), \mu_B(z)] \quad \forall z \in Z \quad (5)$$

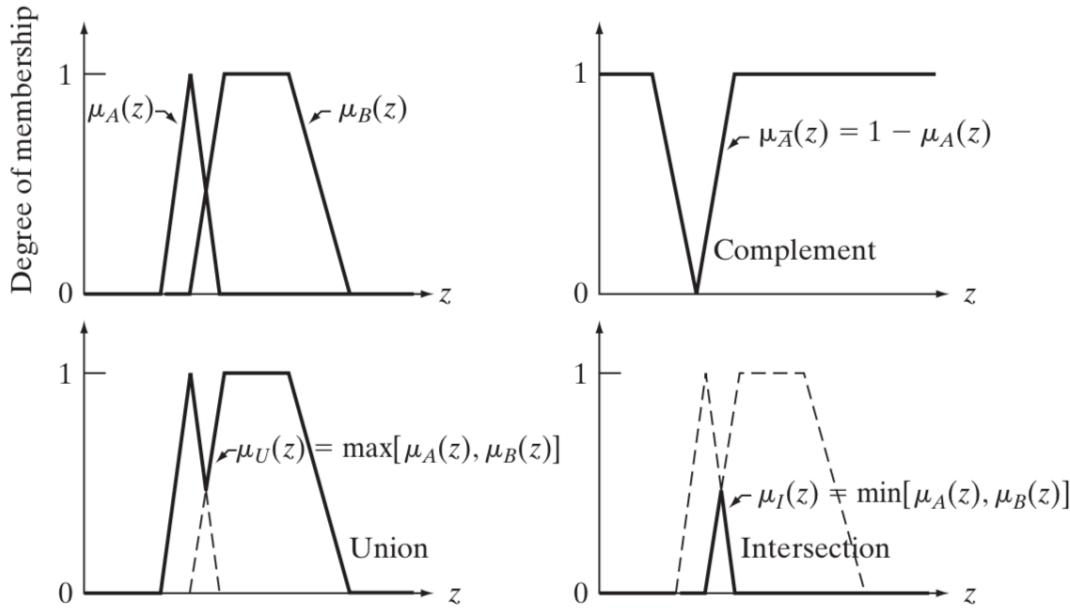


Fig. 2: Funciones de pertenencia de dos conjuntos. Fuente Digital Image Processing de González.

Podemos observar en la gráfica de la esquina superior izquierda la función de pertenencia de dos conjuntos A y B, mientras que en la esquina superior derecha se observa la función de pertenencia del complemento de A y en las dos gráficas inferiores identificamos las funciones de pertenencia de la unión y la intersección del conjunto A y B.

A partir de los principios teóricos que definen las técnicas de fuzzy, nos centramos en su aplicación para alcanzar los objetivos de este proyecto.

Fuzzy presenta principalmente dos aplicaciones en el procesamiento de imágenes, por un lado, filtros espaciales y por otro lado transformaciones de intensidad en una imagen.

2.2 LAPLACIAN OF GAUSSIAN FILTER

En primer lugar, vamos a definir el filtro LOG, que es el que finalmente he utilizado en el programa. Se basa en el uso de la segunda derivada o la laplaciana, que se entiende como la curvatura en todas las direcciones. A diferencia del gradiente, el cual está basado en el uso de la primera derivada, no necesitaremos usar un umbral para detectar los bordes, lo cual resulta una gran ventaja que se une al hecho de que es inmune a la rotación. Este filtro se aplica a partir de la convolución de la imagen con una gaussiana y posteriormente el cálculo de la laplaciana. Otra de las características de este filtro es que recuerda a la forma de ver de los seres humanos, ya que los contornos presentan líneas cerradas. Sin embargo, este filtro no es la panacea, ya que presenta un problema a considerar que es de la localización, esto significa que habrá cierto desplazamiento del punto detectado como borde al punto real.

2.3 GRADIENTE

El gradiente está basado en el operador (primera) derivada o convolución y a partir de él, obtenemos un vector cuyo módulo va a corresponder a la intensidad del borde y el vector que indica la dirección predominante del borde. De forma práctica se entiende el módulo

del gradiente como la suma en valor absoluto del gradiente en x más el de y, lo cual nos evita un elevado coste computacional. Sin embargo, el principal inconveniente es que es muy sensible al ruido, por lo que, sólo analizaremos diferencias locales. Además, para poder identificar el pico en la primera derivada, será necesario establecer un umbral, lo cual supone una gran desventaja. Por el contrario, si empleamos métodos basados en la segunda derivada, lo que debemos identificar es dónde nuestra señal cambia de signo, por lo que no será necesario el uso de un umbral como mostraremos en el filtro LOG. Los inconvenientes por tanto son que los contornos detectados no son de un único píxel de ancho, y debido al ruido no se identifican píxeles pertenecientes a contornos y se detectan otros como tal cuando en realidad no lo son.

Definimos por tanto el gradiente horizontal (g_x) y el gradiente vertical (g_y) mediante las ecuaciones (6) y (7).

$$g_x = \frac{\partial f(x, y)}{\partial x} = f(x + 1, y) - f(x, y) \quad (6)$$

$$g_y = \frac{\partial f(x, y)}{\partial y} = f(x, y + 1) - f(x, y) \quad (7)$$

Cabe destacar que podemos filtrar una imagen $f(x, y)$ con una máscara 1-D a partir de las ecuaciones (6) y (7) para poder obtener los bordes horizontales y verticales de una imagen, pero si además pretendemos detectar los bordes diagonales, tendremos que emplear una máscara de 2-D. De esta forma, tenemos en cuenta que para una región 3x3 de una imagen tendremos unos valores de intensidad generales z tales que:

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

Gradiente Roberts

Está basado en un entorno de vecindad de 2x2. El resultado obtenido con este filtro no es muy bueno, ya que es muy sensible al ruido debido a que presenta una respuesta débil a los bordes porque sólo se tienen en cuenta los píxeles diagonales. Aplicaremos de forma independiente cada uno de los núcleos de convolución y posteriormente asociaremos la información obtenida en una única imagen mediante la suma de resultados de cada filtro. La principal ventaja de este filtro es el bajo coste computacional que presenta, pero podemos observar que el resultado obtenido al filtrar el plato presenta mucho ruido, por lo que no será válido en nuestro caso.

A continuación, se muestra la máscara del filtro Roberts.

-1	0	0	-1
0	1	1	0

Gradiente Prewitt

El operador o gradiente de Prewitt está basado en un entorno de vecindad 3x3, por lo que es menos sensible al ruido que el de Roberts al no depender tanto de las variaciones locales de los niveles de píxel. De esta forma, identifica en uno de los núcleos de convolución las variaciones horizontales y en el otro las verticales. En este caso, variando el valor del umbral o threshold tan sólo podemos identificar correctamente el defecto de nuestro plato si consideramos como borde la sombra del plato, lo cual supone una gran desventaja y se aleja del método de detección que buscamos.

A continuación, se muestra la máscara del filtro Prewitt.

-1	-1	-1	-1	0	1
0	0	0	-1	0	1
1	1	1	-1	0	1

Gradiente Sobel

El gradiente Sobel se basa en el gradiente Prewitt, pero da más valor a los bordes horizontales y a los verticales mediante un valor de dos en los píxeles de conectividad a cuatro y sigue presentando los inconvenientes de las sombras indicados en el gradiente de Prewitt.

A continuación, se muestra la máscara del filtro Sobel.

-1	-2	-1	-1	0	1
0	0	0	-2	0	2
1	2	1	-1	0	1

2.4 CANNY

Uno de los posibles algoritmos a emplear es el detector Canny, el cual presenta un algoritmo más complejo que los filtros anteriores, pero nos permite obtener un resultado más preciso. Dicha precisión se basa en tres premisas destacadas que son las siguientes:

- Minimizar el error de tal forma que todos y únicamente los bordes sean detectados.
- Aumentar la eficiencia de la respuesta de tal forma que para cada píxel real del borde obtengamos únicamente un píxel al emplear el filtro de Canny.
- Reducir el problema de la localización para que los bordes detectados se encuentren en la posición de los bordes de la imagen, es decir, que la distancia entre el borde real y el detectado sea la menor posible.

con lo que conseguiríamos identificar todos y únicamente aquellos píxeles pertenecientes al borde. Este método establece que el método óptimo de cálculo de bordes es la derivada de una gaussiana. El algoritmo se aplica de la siguiente forma:

1. A partir de nuestra imagen I y una gaussiana unidimensional G , derivamos G horizontal y verticalmente con lo que obtendremos G_x y G_y .
2. Convolucionamos I con G_x y G_y , obteniendo así I_x e I_y .
3. Obtendremos el gradiente en módulo y dirección, por lo que podremos eliminar aquellos píxeles que no sean máximos en su dirección y quedarnos así con los verdaderos bordes.

Finalmente aplicaremos una histéresis a partir de dos umbrales, siendo uno de ellos mayor que el otro.

4. PRUEBAS

El método finalmente elegido resultó de realizar numerosas pruebas para identificar qué operaciones y filtros nos permitían obtener un mejor resultado en cuanto a la detección de defectos del plato. A continuación, mostraré los resultados de las pruebas realizadas para la detección de defectos del plato de la figura 3 (a menos que se especifique otro plato).

Cuando comencé a desarrollar el método de detección de defectos de los platos, hice uso de algoritmos tales como la transformada de Hough, cuyo resultado se puede observar en la figura 4.



Fig. 3: Plato parte delantera a detectar defectos.

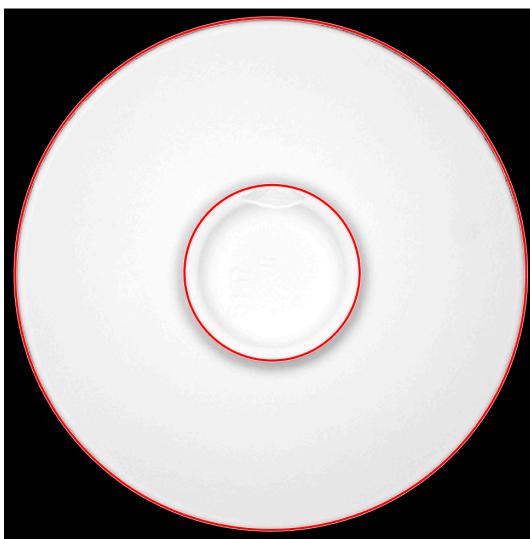


Fig. 4: Resultado imagen al aplicar la transformada de Hough.

Como se puede observar en la figura anterior, en el plato se identifican dos círculos correspondientes a sus bordes que eran casi perfectamente circulares por lo que la detección de estos bordes o círculos era muy exacta. Sin embargo, este método presenta

dos principales desventajas, por un lado, un elevado coste computacional y por otro lado la dependencia de un borde del plato casi perfectamente circular para que pueda ser detectado. Por un lado, para poder analizar el círculo interior identificamos que éste tenga un radio comprendido entre sesenta y noventa centímetros, mientras que hay que realizar un análisis adicional para el círculo exterior y aumentar la sensibilidad a 0.97, característica que no teníamos en cuenta para el borde interior. Por otro lado, aunque podemos apreciar visualmente que son detectados los bordes del plato de la figura, lo que haremos a continuación es dibujar sobre la imagen del plato filtrado mediante el algoritmo LOG dos círculos a partir de las posiciones de los bordes circulares del plato. El problema resulta cuando el defecto se encuentra en el propio borde, ya que en este caso eliminaremos parte del mismo y si se trata de un defecto pequeño, podríamos suprimirlo por completo y por tanto el plato sería correcto y no se desecharía, lo cual no queremos que ocurra.



Fig. 5: Defecto detectado tras aplicar el filtro LOG a la imagen.

Identificamos en la figura 5 cómo una parte del defecto ha sido eliminada tal y como anticipábamos y si reducimos el grosor del círculo situado encima del borde interior del plato, hay partes del mismo que no deberían observarse, ya que no son defecto, pero se detectarían como una parte errónea del plato tal y como se muestra en la anterior figura.

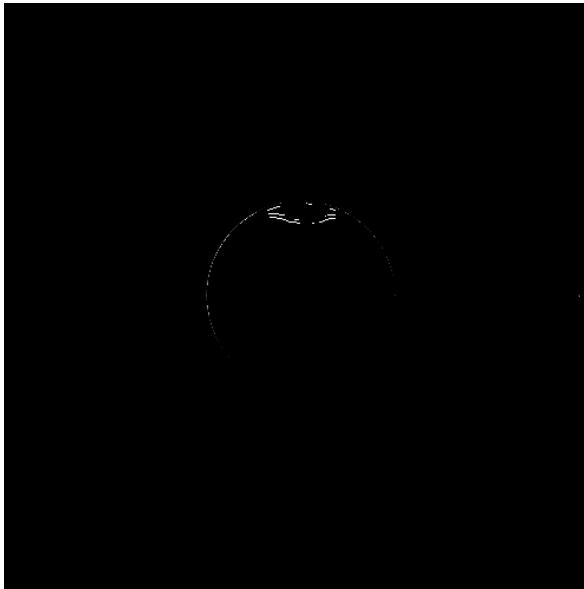


Fig. 6: Detección defecto del plato con resultados erróneos.

A continuación, probaremos a detectar los defectos de un plato con bordes que presentan irregularidades y que no son completamente circulares como es la siguiente figura.



Fig. 7: Plato con defecto en la parte trasera y borde irregular.

Al aplicar la transformada de Hough en la figura anterior, el borde exterior del plato no es detectado ya que no es un círculo perfecto. Podemos asegurar por tanto que la detección de bordes mediante la transformada de Hough no es el método óptimo para este proyecto.

Otro método de detección que empleé al comienzo fue el del número de Euler, como se puede observar en la figura 8. Para el mismo plato en el que se han identificado los bordes en la figura 3 obtenemos un resultado deficiente ya que tan sólo conseguimos identificamos el borde exterior del plato.

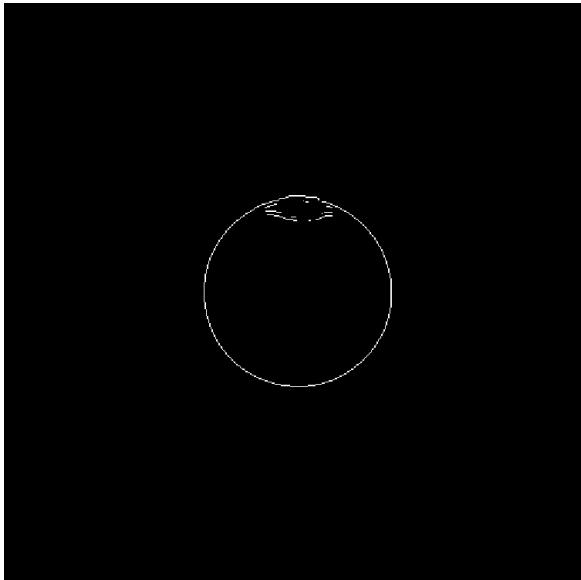


Fig. 8: Detección de bordes fallida con el número de Euler.

Concluimos por tanto que los algoritmos anteriores no resultan eficaces en el proyecto ya que si los platos a analizar no presentan círculos y bordes perfectos los bordes exteriores del plato no son detectados, pero si disminuimos la sensibilidad de detección para que se puedan identificar dichos círculos de los platos, no seremos de capaces de detectar los defectos del plato que tengan unas dimensiones muy pequeñas, los cuales conseguimos detectar con el método finalmente empleado.

Por otro lado, también realizamos pruebas con otros métodos de detección, tal y como la detección de bordes zerocross en la cual aplicamos los mismos pasos al comienzo del programa para redimensionar la imagen, recortarla, centrarla y pasarla a binaria. Posteriormente utilizamos una máscara de Sobel, mediante la cual detectamos los bordes usando el método “zerocross”, de esta forma después haber filtrado la imagen esta operación devuelve los bordes con un valor superior al threshold (en nuestro caso utilizamos 0.02). Observamos en la siguiente figura el resultado de aplicar el filtro a la imagen de la figura 3.

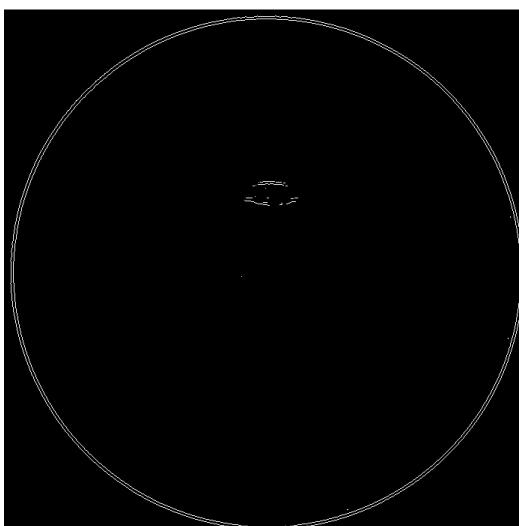


Fig. 9: Detección de bordes zerocross Sobel.

Para poder observar mejor los bordes detectados sobre el plato original, sobreponemos dichas imágenes con lo que obtenemos el resultado mostrado en la figura 10 y 11.

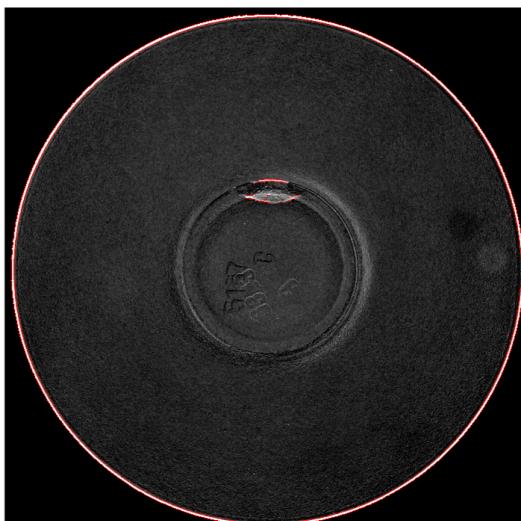


Fig. 10: Defecto marcado imagen binaria.



Fig. 11: Defecto marcado imagen plato.

Observamos que pese a detectar el defecto correctamente, también identificamos el borde exterior del plato el cual no debería de marcarse y un defecto, posiblemente debido al ruido, en el centro del plato, lo cual también ocurre al aumentar el contraste local de una imagen RGB usando localmente el filtro Laplaciano.

Para ello, establecemos los parámetros de filtro adecuados para tan sólo detectar aquellos detalles menores a 0.4. A continuación, aplicamos el llamado filtro rápido y local Laplaciano (locallapfilt) sobre la imagen RGB ya recortada con lo que podemos observar en el montaje de la figura 12 la imagen de entrada (izquierda) y la filtrada (derecha) juntas.

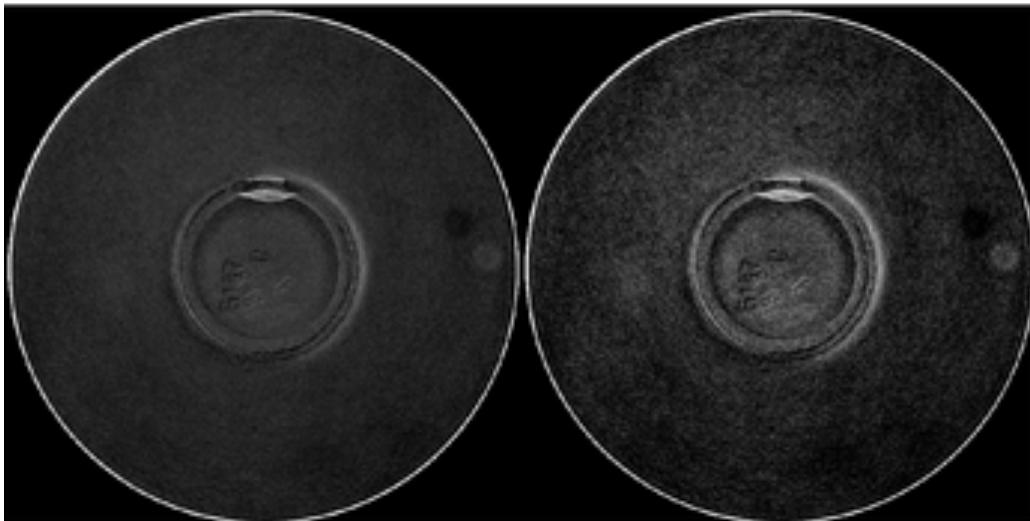


Fig. 12: Montaje imagen original y filtrada mediante el filtro Laplaciano local y rápido.

Posteriormente aplicamos el filtro Sobel y eliminamos aquellos píxeles de la imagen que tienen un tamal menor a cinco para poder evitar ruido.



Fig. 13: Imagen frontal plato bordes detectados.

Finalmente superponemos la figura 3 sobre la figura 13 con lo que obtenemos la siguiente detección de defectos marcada en rojo en la figura 14.



Fig. 14: Resultado detección de defectos filtro Laplaciano local y rápido.

Identificamos que tal y como se producía en los casos anteriores, el borde exterior del plato queda marcado como defecto, lo cual no es aceptable en nuestro caso.

Otro método que también aplicable para la detección de bordes es fuzzy.

En este caso los resultados fueron decepcionantes ya que ni siquiera se consiguió detectar el defecto además de identificar el borde exterior del plato como en los casos anteriores. En cuanto al programa, en primer lugar, la foto tomada del plato era recortada y centraba y posteriormente se aplicaba el filtro Fuzzy. En este caso hemos empleado funciones triangulares ya que obteníamos un resultado superior que si utilizábamos otro tipo de funciones tales como Gaussianas. Para poder aplicar dicho filtro primero convertimos la imagen a float y aplicamos las leyes definidas y posteriormente volvemos a transformar la imagen a su formato original. En la figura 15 observamos el resultado obtenido tras aplicar dicho método de detección de bordes.



Fig. 15: Resultado detección de defectos mediante Fuzzy.

Si modificamos las funciones de pertenencia conseguimos el siguiente resultado.



Fig. 16: Resultado detección de defectos mediante Fuzzy con nuevas funciones de pertenencia.

A partir de la figura 16 podemos identificar que el método de detección de bordes mediante fuzzy se realiza correctamente pero el problema es que en nuestro caso buscamos ir un paso más allá y no identificar los bordes del plato, razón por la cual no nos decantamos finalmente por este método.

Definimos por tanto que debe haber un equilibrio entre las dimensiones de los defectos de los platos (según los estándares de calidad) y la detección de los mismos. Por otro lado, hay que tener en cuenta que el coste computacional del método debía ser bajo y sobre todo dado que se iba a aplicar en tiempo real, no podía retrasar al resto de la producción.

3.1 ELECCIÓN DEL FILTRO

Una vez que tenemos la foto del plato recortada y binarizada, procedemos a aplicar la detección de bordes. Para el cálculo de bordes podremos aplicar diferentes filtros que nos darán diversos resultados, compararemos entonces los resultados obtenidos y elegiremos el filtro que más se adecúe a nuestras necesidades. Para ellos, partimos de la imagen de un plato por su cara frontal y procedemos a analizar los diversos filtros.



Fig. 17: Plato con rotura en el borde interno.

A continuación, muestro los resultados de las diferentes pruebas realizadas con distintas máscaras hasta que llegué a la conclusión de hacer uso del filtro mencionado.

3.1.1 GRADIENTE

Gradiente Roberts

A continuación, se muestra la detección del defecto tras aplicar el filtro Roberts con un umbral de sensibilidad de 0.025.



Fig. 18: Resultado filtro Roberts.

El resultado obtenido presenta ruido en el contorno exterior del plato, aunque a diferencia del método anterior hemos conseguido reducir lo máximo posible sin dejar de detectar el contorno del defecto, el sombreado del plato.

Gradiente Prewitt

A continuación, se muestra el resultado de aplicar la máscara Prewitt con un umbral de sensibilidad de 0.025.

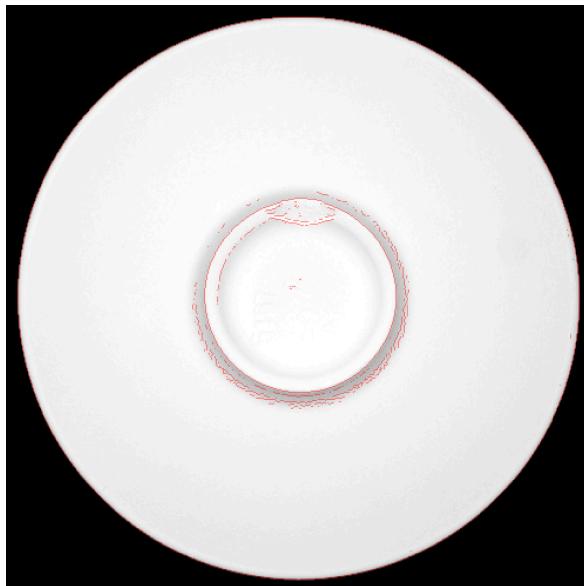


Fig. 19: Resultado filtro Prewitt.

Seguimos observando los mismos problemas identificados en el gradiente anterior en cuanto a ruido.

Gradiente Sobel

En la siguiente figura se muestran los defectos encontrados tras aplicar el filtro Sobel con un umbral de sensibilidad correspondiente a 0.03.



Fig. 20: Resultado filtro Sobel.

A continuación, nos apoyamos también en las transformaciones morfológicas para poder comprobar la eficiencia de este filtro al aplicarlos en otro tipo de defecto.



Fig. 21: Imagen plato parte trasera a filtrar.

Sobre la figura 21 aplicamos el filtro Sobel con un umbral de sensibilidad de 0.04 y posteriormente eliminamos el ruido de tal forma que aplicamos dos transformaciones morfológicas, Closing y Opening con lo que obtenemos el siguiente resultado.

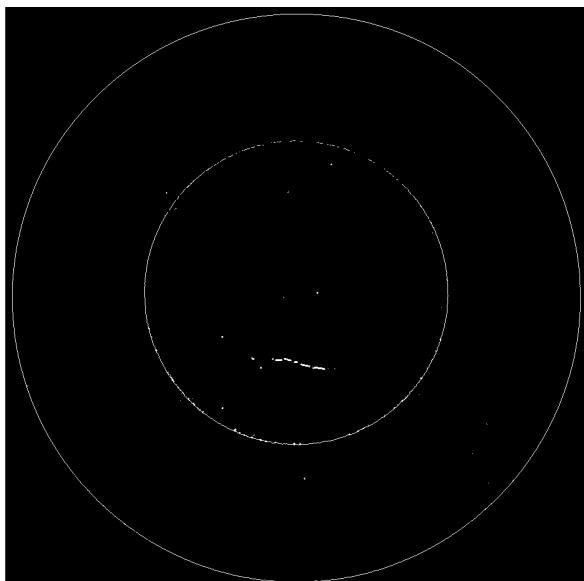


Fig. 22: Imagen plato parte trasera tras aplicar Closing.

Podemos observar en la figura 22 que el defecto no está siendo detectado en su totalidad. A continuación, aplicamos el Opening a nuestra imagen.

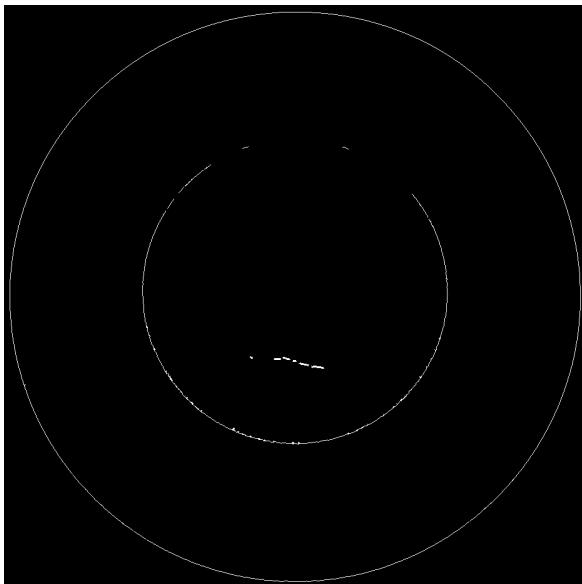


Fig. 23: Imagen plato parte trasera tras aplicar Opening.

En la figura 23 identificamos que al eliminar el ruido de nuestra imagen también hemos conseguido quitar el borde interior del plato, el cual en este caso no nos interesa detectar ya que no presenta ningún defecto.

A continuación, superponemos la imagen del plato original con el filtrado con lo que obtenemos la siguiente detección de defectos.



Fig. 24: Resultado detección de defectos Sobel y operaciones morfológicas.

Pese a que el defecto no se ha detectado en su totalidad, sí que una parte de él ha sido reconocida como tal pero el resultado obtenido sigue presentando el mismo problema que en los casos anteriores de uso de filtros basados en la primera derivada o gradiente y es que continuamos detectando los bordes del plato y el resultado presenta mucho ruido, por lo que para eliminarlo también hemos reducido la intensidad del defecto del plato.

Concluimos, por tanto, que el uso del gradiente para detectar contornos no presenta las soluciones óptimas en nuestro caso, por lo que desecharemos este método.

3.1.2 CANNY

Como podemos observar en la siguiente figura, mediante el método de detección de bordes Canny identificamos hasta los contornos de la sombra del plato, cuya información no nos es relevante.



Fig. 25: Resultado filtro Canny.

3.1.3 LAPLACIAN OF GAUSSIAN FILTER

El filtro LOG es el que finalmente he utilizado en el programa debido a los defectos detectados tras emplearlo.



Fig. 26: Resultado filtro LOG.



Observamos que a diferencia de con los filtros anteriores, ahora identificamos únicamente los bordes interior y exterior del plato y el defecto del mismo sin la presencia de ruido, por lo que concluimos que este método es el mejor.



4. DISEÑO E IMPLEMENTACIÓN

A continuación, explicaremos la ejecución del programa desde las etapas de detección del defecto del plato, los algoritmos adicionales empleados y el interfaz gráfico diseñado.

4.1 TRANSFORMACIONES MORFOLÓGICAS

Por un lado, nos apoyamos en las operaciones morfológicas para poder determinar si el plato en cuestión presenta algún roce o le falta la capa superior del mismo debido a un golpe, no significando esto que falte un trozo de plato. Lo que hacemos en primer lugar es un closing, es decir, una dilatación seguida de una erosión. Con la dilatación conseguimos gráficamente aumentar el número de píxeles del borde, bajo los que cae el elemento estructural, el cual se define como el conjunto de puntos o patrón que nos permite determinar la estructura de nuestra imagen. Posteriormente se realiza una erosión, también definida como transformación dual de la dilatación, en el cual gráficamente eliminaremos aquellos píxeles que caen bajo el elemento estructural. Finalmente aplicaremos la transformación morfológica top-hat, que consiste en restar a la imagen su opening.

4.2 RESTA DE IMÁGENES

Dado que tenemos una imagen del plato correcto, lo que haremos será restar dicha imagen a la del plato entrante y obtener así las diferencias entre ambas. Dado que los platos no son exactamente iguales y no presentan bordes lisos, al aplicar este método tendremos algunos falsos positivos, pero esto podremos controlarlo con el establecimiento de umbrales de correlación de tal forma que no eliminemos platos correctos.

4.3 ELIMINACIÓN DE RUIDO

Una vez realizadas estas operaciones, lo que habremos conseguido es eliminar el defecto de la imagen del plato por lo que a continuación tendremos que restar la imagen del plato con el defecto a esta última obtenida mediante los algoritmos mencionados. Para poder eliminar el ruido de nuestra imagen aplicaremos el comando *bwareaopen*, el cual permite eliminar todos aquellos componentes conectados que tengan, en nuestro caso, menos de diez píxeles en la imagen binaria y produciendo así una nueva imagen binaria.

4.4 ANÁLISIS DEL DEFECTO

El siguiente paso consiste en obtener el tamaño de dicha imagen binaria y recorrerla, de tal forma que si hemos encontrado los tipos de defectos mencionados anteriormente (roce o descascarillamiento), se habrá quedado marcado el defecto en la imagen binaria mediante píxeles blancos y esto significará que hemos realizado una correcta identificación y que el plato es defectuoso, mientras que en el caso de no encontrar dichos píxeles, realizaremos un segundo test para descartar otro tipo de defectos como roturas en el plato.

Como ya he comentado anteriormente, para el correcto funcionamiento del programa será necesario disponer de una iluminación y una distancia entre la cámara y la cinta

transportadora por la que pasan los platos, constante, ya que en caso contrario tendremos falsos positivos y la fábrica perderá dinero y retrasará la producción. Finalmente, la interfaz gráfica el programa se presenta del siguiente modo mostrado en la figura 27.



Fig. 27: Interfaz programa detección de defectos en artículos de cerámica.

En las gráficas superiores mostraremos el plato correcto por su cara delantera y el resultado del análisis realizado del plato entrante. De igual manera en la fila de abajo mostraremos el plato correcto por detrás y a continuación los resultados obtenidos a partir del plato entrante. Según el tipo de plato seleccionado, se mostrará en la parte superior de la pantalla el correspondiente defecto del plato.

A continuación, muestro mediante un diagrama de flujo el proceso de detección de defectos en los artículos de cerámica dentro de la fábrica y sus distintas fases:

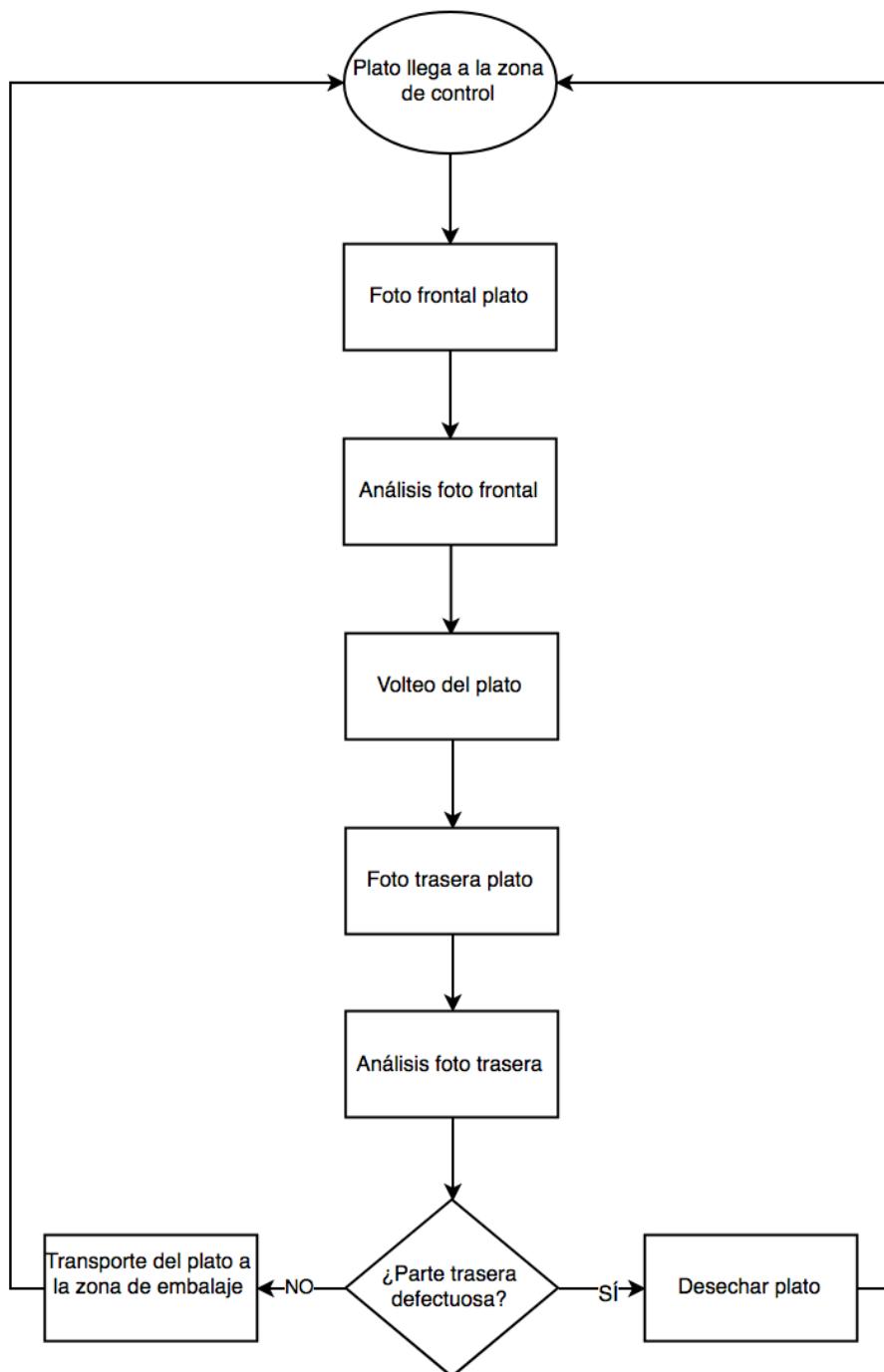


Fig. 28: Diagrama de flujo del proceso de detección de defectos en la fábrica.

Este proyecto se centra en la fase de análisis de las fotos tomadas por una cámara a ambas caras del plato.

El primer problema con el que me encontré fue el de tener que identificar diferentes defectos en los platos, los cuales podían ser roturas o simples arañazos en ellos. El programa por desarrollar no debía suponer un elevado coste computacional y tenía que ser capaz de dejar pasar unívocamente a aquellos platos correctos de tal forma que tan sólo elimináramos los que presentan algún tipo de defecto, sin importarnos el tipo. En mi caso decidí crear una función a la cual pasar las imágenes de un plato correcto y la de un plato pasando por la cinta transportadora en la fábrica y realizar dos test diferentes.

En primer lugar, disponemos de dos imágenes que corresponden a un mismo modelo de plato, una que ha pasado unos exhaustivos controles de calidad y se ha definido como plato correcto, y otra que corresponde a la imagen obtenida del plato que circula por la cinta transportadora y hemos de clasificar. Comenzamos primero por analizar la parte frontal del plato y posteriormente la trasera, de tal forma que, si una de las dos presenta algún defecto, el plato será desecharido y en caso contrario éste pasará a la siguiente fase de embalaje en la fábrica.

Al comenzar, recortaremos las imágenes, lo cual no será necesario en una aplicación práctica, ya que como hemos comentado tendremos un entorno controlado y una cámara fija. Una vez ajustados los tamaños de las imágenes, de tal forma que coincidan sus centros, transformaremos la imagen a binaria. Finalmente, procederemos a aplicar un filtro, en nuestro caso el “Laplacian of Gaussian filter” también conocido como LOG.

Los comandos utilizados en Matlab para realizar dichos pasos son, por un lado, *imresize* y *rgb2gray* empleados inicialmente para modificar la resolución de la imagen original y transformarla en una imagen binaria respectivamente. Después realizaremos un *closing* (transformación explicada posteriormente) y dado que tenemos una imagen de tipo lógica, la convertiremos en entera de 8 bits y sin signo (*uint8*). Posteriormente tan sólo nos quedaremos con aquellos píxeles mayores de veinte y obtendremos así una imagen de tipo lógica que usaremos para obtener el centro y las distancias de los ejes en la misma (*regionprops*), lo cual nos servirá para recortar la imagen y centrarla. El siguiente paso será aplicar a la imagen correcta la detección de bordes mediante el filtro LOG.

Para la imagen a analizar, realizaremos los mismos pasos, pero en este caso centraremos la imagen según la del plato correcto para poder evitar que los bordes de los platos no coincidan y obtener falsos positivos. Finalmente aplicamos el filtro LOG.

A continuación, las dos imágenes, aquella a la que hemos aplicado el filtro y la correspondiente al plato original y correcto, pasarán como parámetros a la función creada, la cual en primer lugar realizará una transformación morfológica de tipo *closing*, seguida por una *Top-Hat*, obteniendo como resultado las diferencias entre la imagen defectuosa y la resultante tras aplicar dichas transformaciones. Posteriormente utilizaremos el comando *bwreaopen* para poder eliminar aquellos pequeños objetos que tengan un tamaño inferior a veinte píxeles y así poder identificar únicamente los defectos y no ruido en la imagen. Comprobaremos que en la imagen obtenida existan píxeles blancos, en cuyo caso mostraremos la imagen original superpuesta por dicho resultado mediante el comando *imoverlay*, y en caso contrario obtendremos la diferencia entre las imágenes obtenidas como parámetros en nuestra función y aplicaremos el comando *bwreaopen*, esta vez eliminando aquellos objetos con un tamaño inferior a cinco píxeles y superponemos la imagen obtenida sobre la correcta.

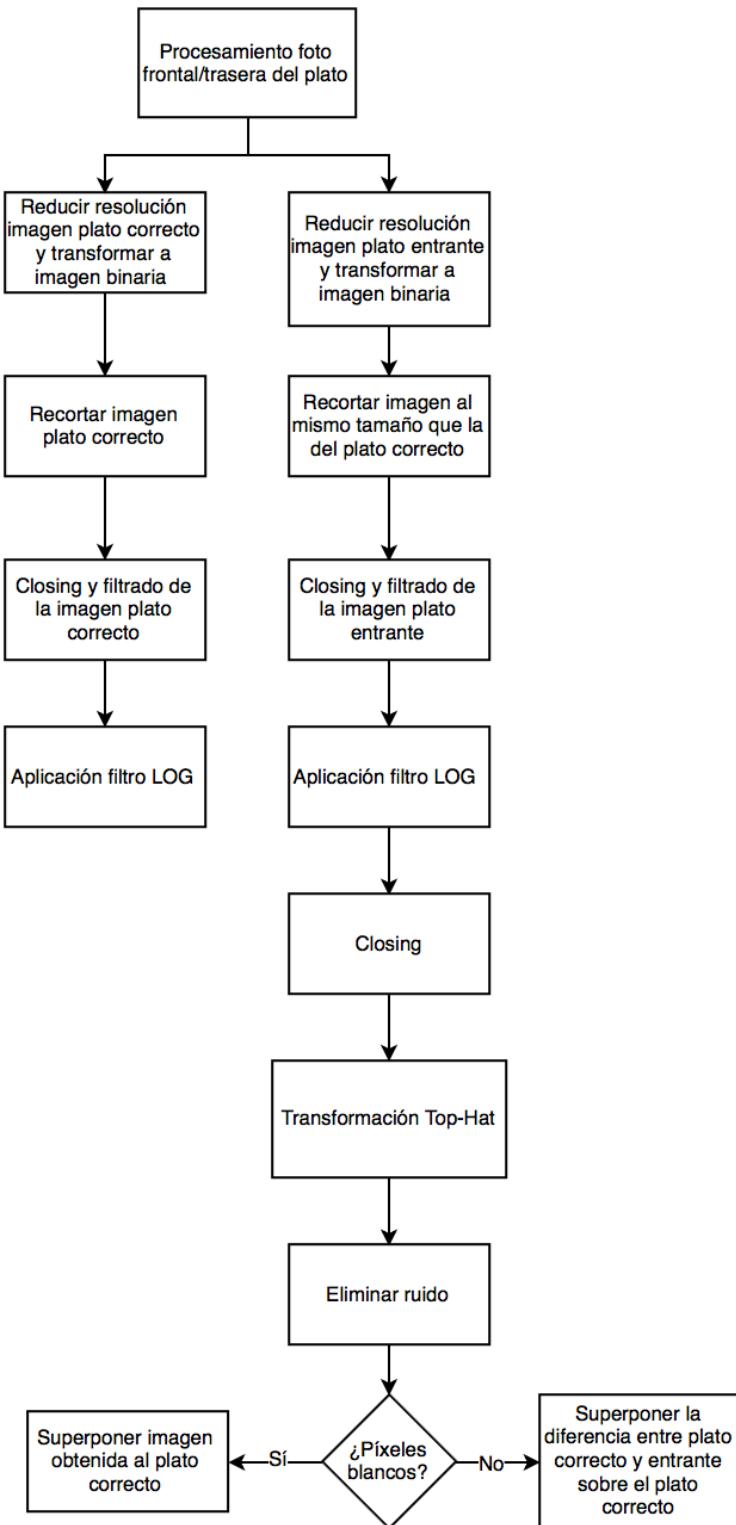


Fig. 29: Diagrama de flujo análisis imagen.

5. RESULTADOS

A continuación, mostraré los resultados obtenidos tras aplicar cada método para cada uno de los defectos.

5.1 ROTURA PARCIAL

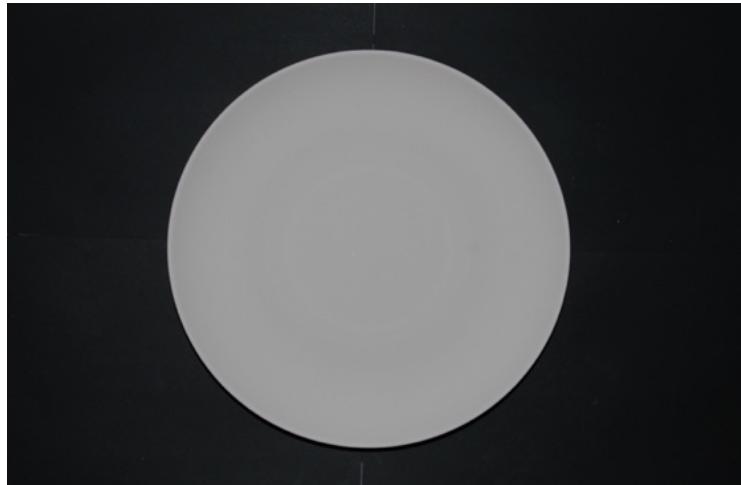


Fig. 30: Frontal plato correcto tipo I.



Fig. 31: Trasera plato correcto tipo I.

El plato por analizar es plano y la parte frontal del mismo (Ilustración 2.7) presenta un círculo interno poco marcado o diferenciado del resto del plato, lo cual lo tendremos en cuenta a la hora de analizar el plato entrante por la cinta, ya que éste no nos interesará.

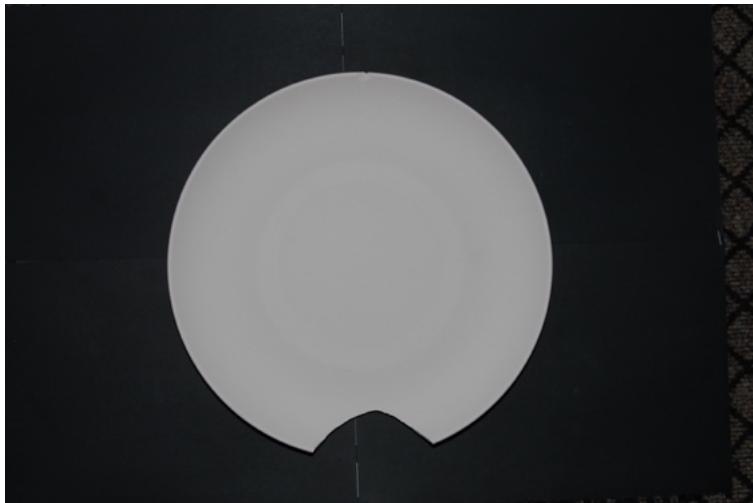


Fig. 32: Foto frontal para analizar tipo 1.



Fig. 33: Foto trasera para analizar tipo 1.

Como podemos observar, el plato presenta una rotura muy significativa, fácilmente identificable en las ilustraciones anteriores, en la parte externa del mismo por lo que deberá ser desecharlo al presentar un defecto considerable. En este caso nos interesará detectar el borde exterior del plato.



Fig. 34: Foto frontal para analizar binaria tipo I.



Fig. 35: Foto trasera para analizar binaria tipo I.

A partir de estas imágenes obtenemos otra binaria y recortada de igual tamaño para cada una de ellas, lo cual va a permitir que los centros coincidan y por tanto los platos también, por lo que podremos operar con las imágenes obtenidas e identificar el defecto.

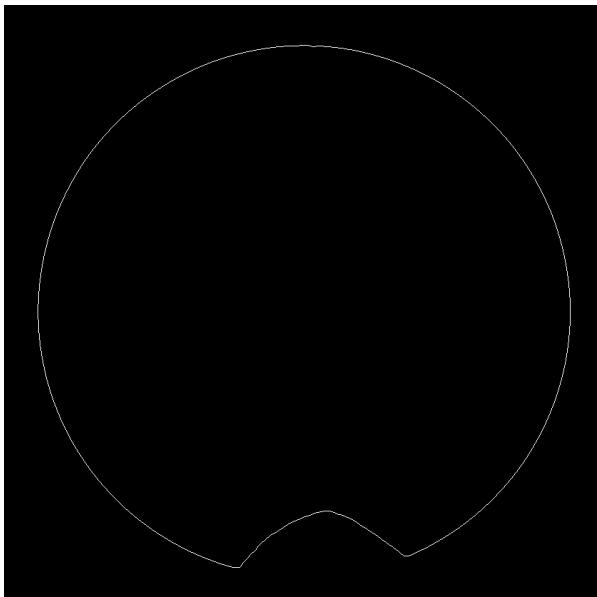


Fig. 36: Foto frontal para analizar filtrada tipo 1.

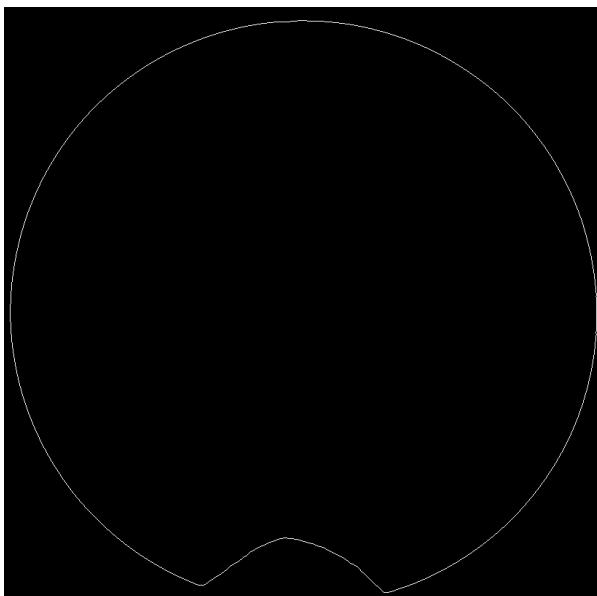


Fig. 37: Foto trasera para analizar filtrada tipo 1.

Tras aplicar el filtro LOG, con los correspondientes valores a cada imagen, obtenemos los bordes de nuestras imágenes y posteriormente eliminamos aquellos píxeles considerados como ruido, es decir, píxeles aislados que toman un nivel de gris diferente al de sus vecinos y que estaríamos considerando como borde cuando en realidad no lo son. Observamos que en este caso el defecto más significativo es la rotura del plato, por lo que nos centramos en él.



Fig. 38: Imagen frontal resultado tipo 1.



Fig. 39: Imagen trasera resultado tipo 1.

Mostramos al usuario, en rojo, las diferencias encontradas entre ambos platos. Dada la gran discordancia entre los resultados obtenidos, podemos afirmar que el plato que hemos analizado se trata de un plato con irregularidades y defectos y no admisible a venta. El hecho de que los bordes exteriores de los platos no coincidan, se deben a que las imágenes han sido tomadas por una cámara situada a diferentes distancias en cada caso.

5.2 ROTURA EN EL BORDE



Fig. 40: Foto frontal plato correcto tipo 2.



Fig. 41: Foto trasera plato correcto tipo 2.

A continuación, disponemos de un plato plano pero de un tamaño superior al tipo 1 analizado anteriormente. En este caso, el borde interior que apreciamos en la figura 43 presenta una mayor importancia que en el caso anterior al disponer de una rotura en el mismo.

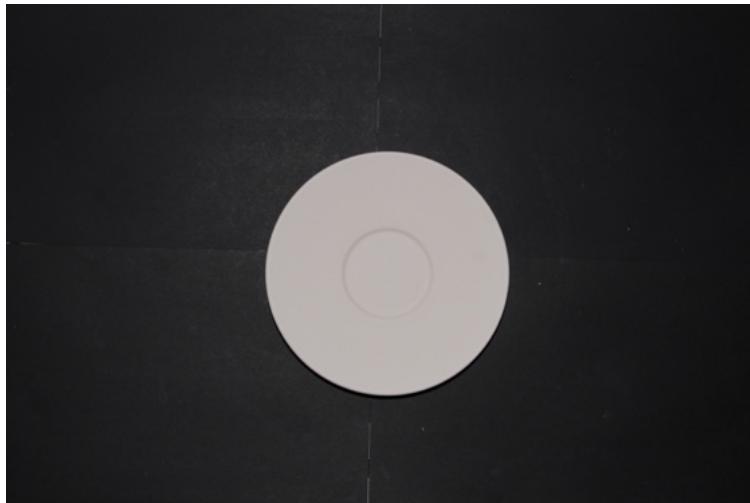


Fig. 42: Foto frontal para analizar tipo 2.

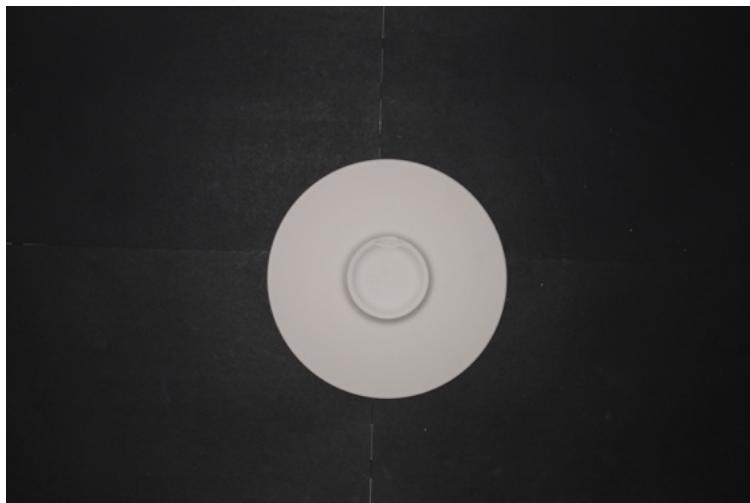


Fig. 43: Foto trasera para analizar tipo 2.

El plato que vamos a analizar presenta una rotura en el borde interno del mismo, la cual puede deberse a un choque entre platos, pero observaremos mejor los posibles defectos del plato en las siguientes ilustraciones, obtenidas tras recortar nuestras imágenes al mismo tamaño y posteriormente centrarlas.



Fig. 44: Foto frontal para analizar binaria tipo 2.



Fig. 45: Foto trasera para analizar binaria tipo 2.

Como ya podíamos observar anteriormente, en la figura 42 comprobamos que la parte trasera del plato presenta un descascarillamiento del borde, mientras que la parte delantera no presenta ningún defecto a simple vista, pero lo comprobaremos tras aplicar el filtro.

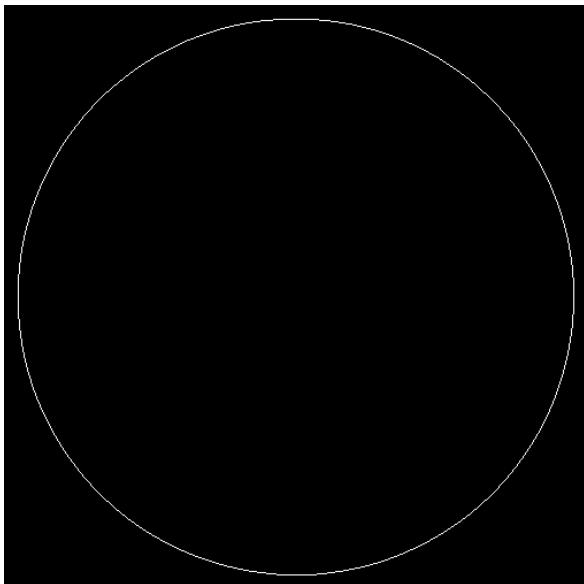


Fig. 46: Foto frontal filtrada para analizar tipo 2.

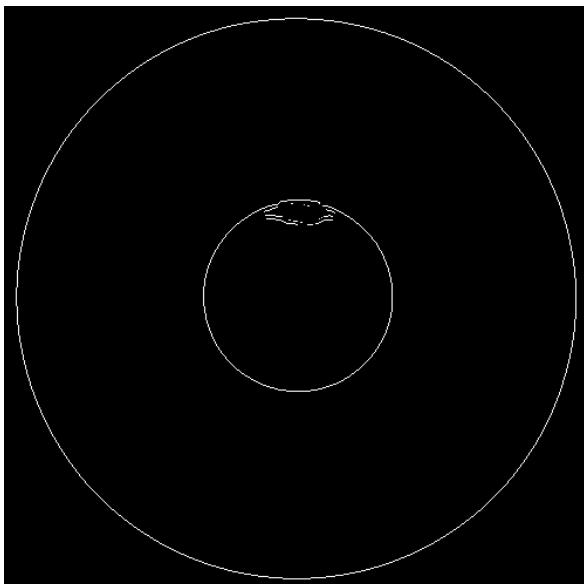


Fig. 47: Foto trasera filtrada para analizar tipo 2.

Aplicamos el filtro LOG a cada una de las imágenes para poder identificar los bordes del plato, así como los posibles defectos existentes. En este caso sí que querremos detectar el borde interno del plato, ya que es ahí donde se encuentra la rotura del mismo.



Fig. 48: Imagen frontal resultado tipo 2.



Fig. 49: Imagen trasera resultado tipo 2.

Finalmente mostramos al usuario que en la parte trasera del plato a analizar se ha detectado una rotura en el círculo interior interno. Por otro lado, dado que los platos no son completamente iguales y que los bordes se encuentran escalonados, también se han detectado pequeñas diferencias entre el plato correcto y el plato que hemos analizado frontalmente, lo cual a simple vista observamos que no es cierto, pero si comprobamos los valores de correlación entre ambos platos, podremos identificar que éste tiene un valor muy superior al que tenemos en caso de aplicar este algoritmo a la parte trasera de nuestro plato, por lo que podríamos establecer un umbral para evitar detectar falsos positivos.

5.3 DESCASCARILLAMIENTO DEL BORDE



Fig. 50: Foto frontal plato correcto tipo 3.



Fig. 51: Foto trasera plato correcto tipo 3.

El artículo tipo 3, corresponde a un plato hondo, cuyo borde interior de la parte frontal del mismo no se puede apreciar, lo cual tendremos en cuenta para detectar los bordes del plato.



Fig. 52: Foto frontal para analizar tipo 3.



Fig. 53: Foto trasera para analizar tipo 3.

A simple vista podemos identificar una marca o arañazo en la figura 52, por lo que para identificar este defecto nos centraremos en analizar el borde interior del plato.

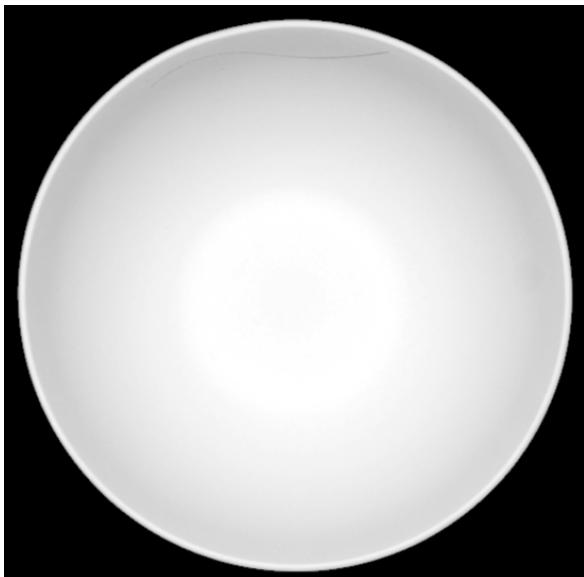


Fig. 54: Foto frontal para analizar binaria tipo 3.



Fig. 55: Foto trasera para analizar binaria tipo 3.

Lo más significativo de la imagen binaria y recortada que obtenemos es que el borde exterior del plato por la zona trasera se encuentra escalonado lo cual nos lleva a pensar que al obtener la diferencia entre la imagen correcta y la imagen a evaluar, obtendremos muchas zonas dispares, de tal forma que tendremos que hacer uso de otros parámetros como la correlación para establecer un umbral de tal manera que no confundamos un plato con defecto a una mala resolución de la imagen de dicho plato. Por otro lado, en la parte frontal identificamos un arañazo en el plato y en la zona trasera, parece que el plato ha sufrido un pequeño golpe en el borde interior del mismo que identificamos con un punto circular.

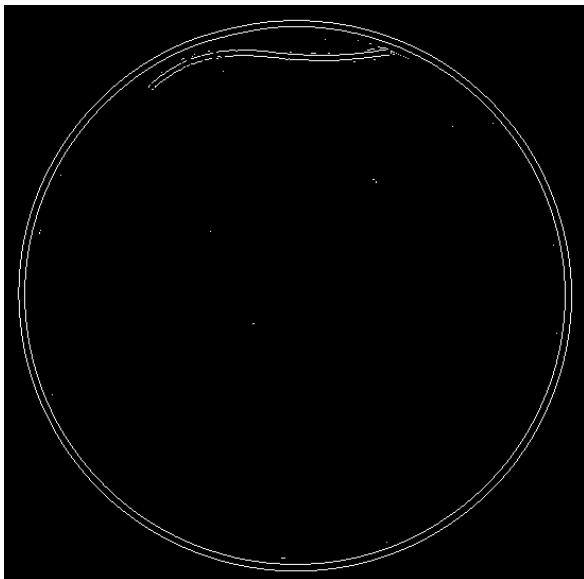


Fig. 56: Foto frontal para analizar binaria tipo 3.

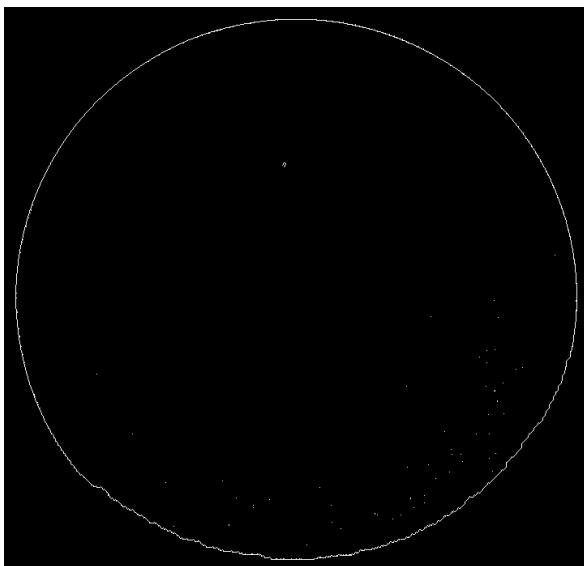


Fig. 57: Foto trasera para analizar binaria tipo 3.

A continuación, aplicamos el filtro LOG para identificar los bordes. Podemos observar cómo se identifican muchos píxeles pertenecientes al borde cuando en realidad son ruido, pero esto lo solucionaremos en el siguiente paso. En cuanto a la cara frontal del plato, debido al juego de luces y sombras de la imagen original, identificamos dos bordes concéntricos que corresponden al borde exterior del plato, sin embargo eso no resulta un problema en este caso ya que conseguimos identificar el contorno del defecto al aplicar el filtro.

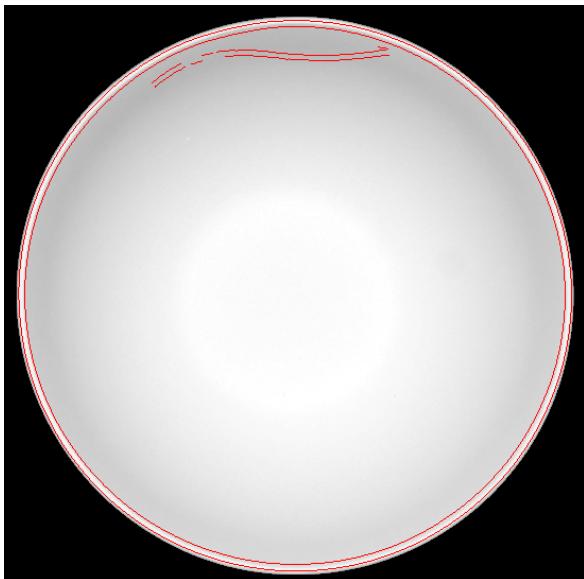


Fig. 58: Imagen frontal resultado tipo 3.



Fig. 59: Imagen trasera resultado tipo 3.

Finalmente mostramos el resultado al usuario, de tal forma que se puede apreciar que en la parte frontal del plato se ha identificado el arañazo mencionado anteriormente y además los bordes de los platos no coinciden, lo cual se puede deber a que las imágenes han sido tomadas desde diferentes distancias o con distintas resoluciones. En cuanto a la parte posterior del plato, hemos conseguido eliminar la mayor parte del ruido identificado al aplicar la detección de bordes y aislar el defecto, el cual se puede apreciar con claridad en los resultados que se muestran.

5.4 GOLPE PARTE TRASERA

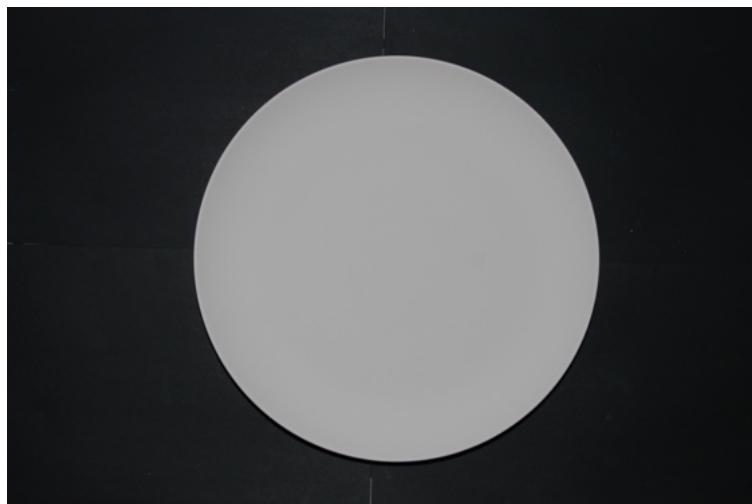


Fig. 60: Foto frontal plato correcto tipo 4.



Fig. 61: Foto trasera plato correcto tipo 4.



Fig. 62: Foto frontal para analizar tipo 4.



Fig. 63: Foto trasera para analizar tipo 4.

En el plato que hemos de analizar, observamos que, en la parte delantera del mismo se identifica un pequeño roce, mientras que en la parte trasera el borde interior presenta una ligera rotura y también identificamos dos roces en la superficie comprendida entre el círculo interior y exterior del plato.

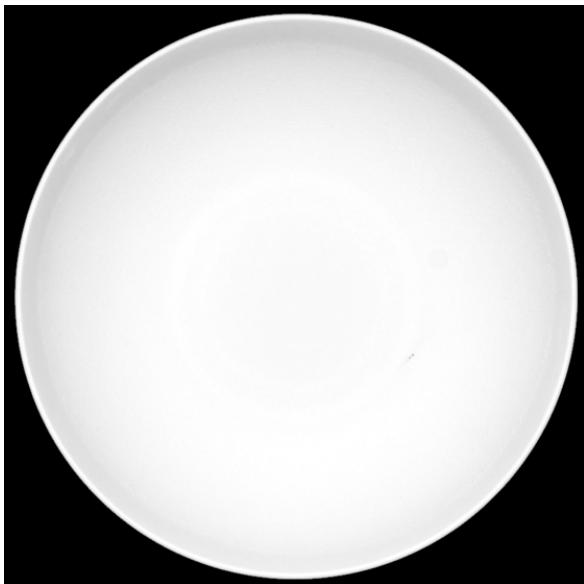


Fig. 64: Foto frontal para analizar binaria tipo 4.



Fig. 65: Foto trasera para analizar binaria tipo 4.

En las ilustraciones anteriores, se puede observar con mayor claridad los defectos del plato, tanto en la parte frontal como trasera del mismo.

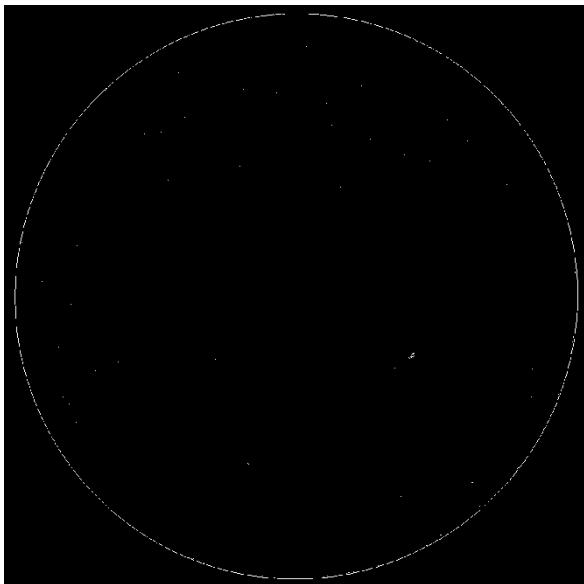


Fig. 66: Foto frontal para analizar filtrada tipo 4.

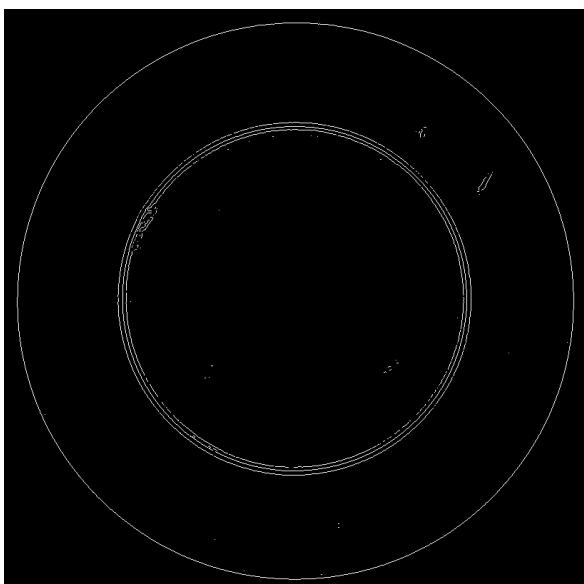


Fig. 67: Foto trasera para analizar filtrada tipo 4.

Después de aplicar el filtro LOG para identificar los bordes de nuestro plato, podemos observar que se han identificado un elevado número de píxeles que se reconocen como ruido y que nos llevarían a confusión a la hora de obtener las diferencias entre los platos. Además, los píxeles identificados en la zona central de la imagen trasera del plato corresponden a las letras empleadas en la fábrica que hacen alusión al modelo de plato y que no todos los platos presentarán las mismas, por lo que no nos interesaría tenerlas en cuenta durante nuestro análisis.



Fig. 68: Imagen frontal resultado tipo 4.

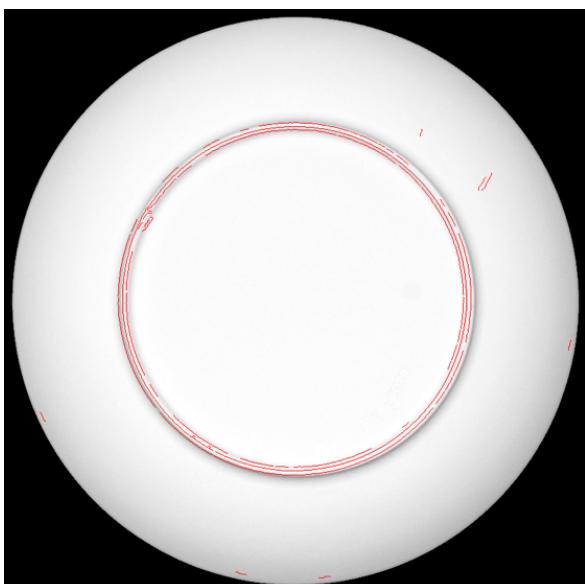


Fig. 69: Imagen trasera resultado tipo 4.

Finalmente mostramos los resultados obtenidos, en los que, en el caso de la parte frontal del plato, hemos conseguido identificar el defecto de forma precisa pese a algunos píxeles del borde del plato que no coinciden en ambos casos. En cuanto a la zona trasera del plato, identificamos los araños mencionados anteriormente y la rotura en el borde, además de observar que el borde interior del plato no coincide con el del plato modelo o correcto, lo cual se puede deber a la sombra del círculo interior del plato sobre el mismo.

5.5 PLATO SIN DEFECTOS

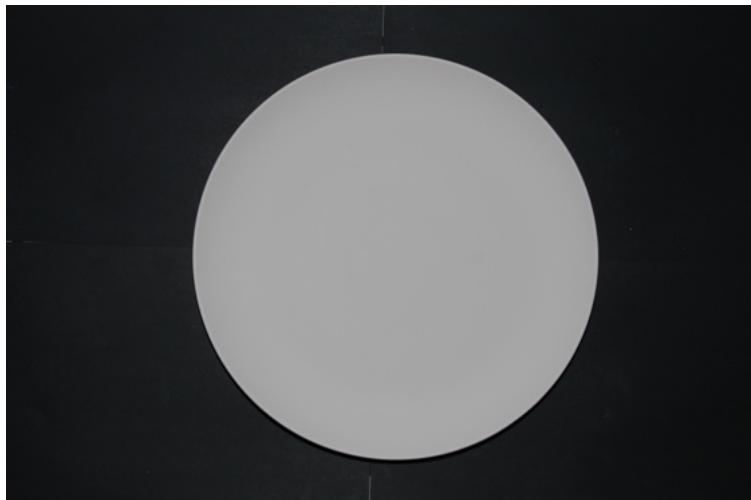


Fig. 70: Foto frontal plato correcto tipo 5.



Fig. 71: Foto trasera plato correcto tipo 5.

En este último caso vamos a identificar un plato que no presenta ningún defecto y por tanto no deberá ser desecharido de la línea de envasado.

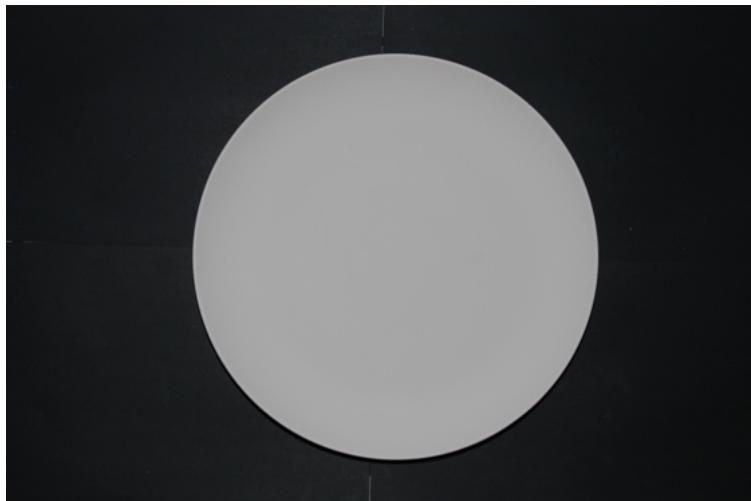


Fig. 72: Foto frontal para analizar tipo 5.



Fig. 73: Foto trasera para analizar tipo 5.

Visualmente no podemos apreciar ningún defecto en el plato.



Fig. 74: Foto frontal analizar binaria tipo 5.



Fig. 75: Foto trasera para analizar binaria tipo 5.

Tras obtener la imagen binaria y recortada del plato que hemos de analizar, corroboramos que no presenta defecto de ningún tipo.

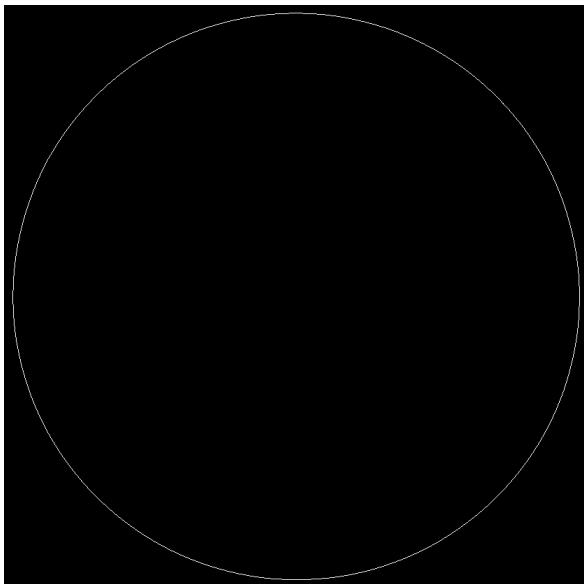


Fig. 76: Foto frontal para analizar filtrada tipo 5.

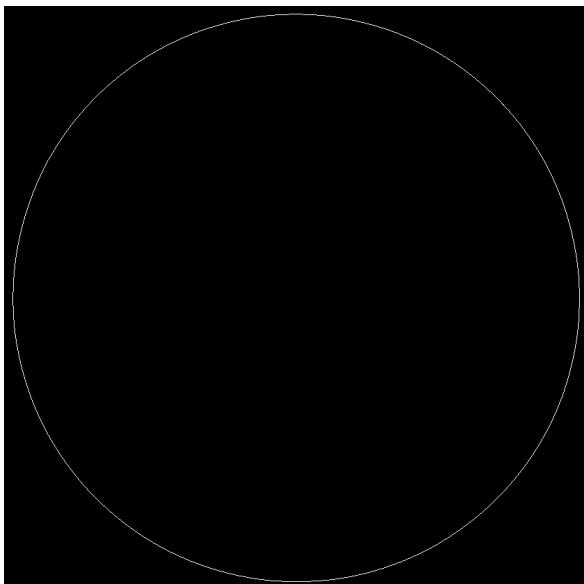


Fig. 77: Foto trasera para analizar binaria tipo 5.

En este caso, tras aplicar el filtro LOG conseguimos que el ruido presente en la imagen sea nulo. En este caso no nos interesa analizar el borde interior del plato porque no presenta ningún defecto, por lo que buscaremos reducir el ruido a costa de detectar dicho borde.

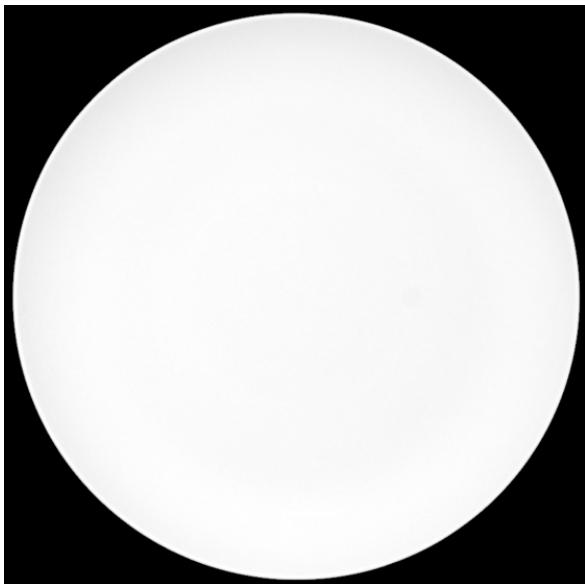


Fig. 78: Imagen resultado frontal tipo 5.



Fig. 79: Imagen resultado trasero tipo 5.

Los resultados obtenidos coinciden con los esperados ya que la imagen del plato se había tomado a la misma distancia y con una iluminación constante en el caso del plato correcto y el plato a analizar, lo cual es una muestra más de la importancia de mantener estos factores constantes para poder analizar correctamente los platos que son transportados por la cinta en la fábrica.

6. CONCLUSIONES

Tras observar los resultados obtenidos, cabe destacar que el principal factor que determinará el éxito del programa será disponer de un entorno controlado con una iluminación constante y una cámara fija. Tal y como esperábamos, los resultados no son perfectos, pero permiten discernir entre un plato correcto y uno defectuoso sin la necesidad de que un operario tenga que comprobarlo, sin embargo, disponer de un mayor número de imágenes de platos correctos y otros posibles defectos, aumentaría la fiabilidad del programa.

Para determinar si el plato entrante por la cinta transportadora presenta un defecto o no, hemos realizado dos tests o evaluaciones, en primer lugar, aplicamos una sucesión de transformaciones morfológicas (closing y transformación top-hat) y comprobamos si el defecto se debe a que se ha levantado una capa del plato, en caso negativo restamos ambas imágenes. Sin embargo, cabe destacar que el programa se ha adecuado a las imágenes de las que se disponía y por tanto a dichos defectos, es decir, según el número y tipo de posibles defectos que queramos abarcar, habrá que aumentar el número de pruebas a realizar para poder mejorar la eficacia del programa.

7. TRABAJO FUTURO

El siguiente a paso a dar sería implementar en Simulink el sistema necesario para dar la orden al robot situado en la fábrica de desechar o no el plato entrante, en función del valor de correlación obtenido al analizar el plato. Además, habría que aumentar el número de imágenes a analizar de tal forma que se pudieran captar un elevado porcentaje de los defectos en los platos y, por otro lado, habría que analizar el canto o borde de los mismos. Finalmente, habría que tomar las fotos de prueba con una resolución mayor para que los bordes de los platos no estén escalonados y se puedan obtener unos resultados más precisos.

Otra mejora significativa sería la de modificar el método a usar, ya que, si se emplea deep learning, conseguiríamos que el propio sistema aprendiese de los errores detectados y cada vez resultaría un método más fiable y con menor posibilidad de fallo. Uno de los problemas de este método resulta en que hay que entrenar el sistema de antemano y habrá que incluir el mayor número de posibles defectos en los platos ya que ante una situación atípica no habrá definida la acción que ha de tomar el sistema, por lo que aparte del coste inicial de programación, hay que incluir el de posibles fallos al comienzo de uso del método, sin embargo, la eficacia aumentará de forma exponencial y será un sistema mucho más preciso que cualquier otro ya que aprenderá con cada plato, obteniendo así las características necesarias para actuar con el siguiente.

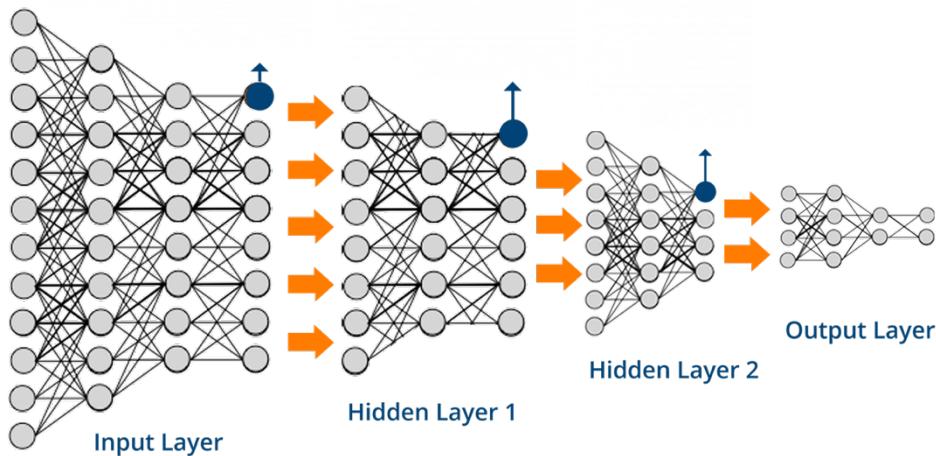


Fig. 80: Ejemplo de detección mediante deep learning.

8. ANEXO

8.1 ASPECTO SOCIOECONÓMICO

La realización de este proyecto surge a partir de la acusada necesidad de aumentar la calidad de los productos dirigidos a los clientes, en este caso nos centramos en la industria de la cerámica y la fragilidad de los productos creados con este material.

En primer lugar, la empresa Ipec está situada en Alba Iulia, zona muy tradicional en Rumania. La empresa cuenta con un contrato exclusivo con IKEA, de tal forma que se dedica a la fabricación de artículos de cerámica tales como platos. Dado que el nivel de desarrollo del país no es elevado, resulta interesante que empresas como Ipec, la cual se encuentra completamente automatizada con más de trescientos robots, le dé este valor añadido a la zona.



Fig. 81: Fábrica Ipec.

La empresa en cuestión actualmente dispone de varios trabajadores que realizan intensivos turnos de una hora (tras los cuales deben cambiar de tarea ya que resulta agotador), para poder reducir los fallos provocados por el cansancio ocular, lo cual supone para la empresa un coste notable, ya que la identificación de defectos en los artículos no se realiza de forma continua por lo que hay franjas horarias en las que el resto de máquinas de la fábrica se paran porque no hay un trabajador 24 horas al día para poder verificar la calidad de los productos. Por otro lado, se trata de un proceso muy repetitivo y cansado para los empleados, ya que comprobar que miles de platos parecidos no tengan ningún defecto, resulta muy pesado y algunos trabajadores acaban con problemas médicos oculares. El objetivo de automatizar este proceso de detección de defectos es aliviar trabajo a los empleados y por consiguiente que éstos puedan realizar una tarea de otro

ámbito, lo cual está respaldado por un aumento notable de trabajadores en la empresa durante los últimos años, actualmente cuenta con alrededor de ochocientos empleados.



Fig. 82: Fábrica Ipec durante revisión manual de defectos.

Automatizar el proceso de detección de defectos permite producir y controlar el estado de cada artículo de forma continua y sin necesidad de parar la producción, así como reducir los fallos humanos y el tiempo de detección de defectos.

8.2 MARCO REGULADOR

Dado que la fábrica de platos en Rumani Ipec ya se encuentra completamente automatizada, la incorporación del método de detección definido (cámara y brazo robótico que dé la vuelta al plato) supondrá necesario aumentar las condiciones de seguridad de la fábrica para evitar que los operarios sufran algún tipo de accidente. Será necesario incorporar una jaula de seguridad alrededor de la zona de detección la cual no podrán traspasar los empleados, así como un botón de parada de emergencia.

En cuanto a los estándares técnicos, para desarrollar el programa he empleado Matlab con la licencia de estudiante que dispone la universidad, cuyo lenguaje de programación es propio, pero guarda gran similitud con otros como C o C++. Por otro lado, para poder verificar la eficacia del método he hecho uso de fotos tomadas in situ en la fábrica de Rumania que me han permitido mediante prueba y error desarrollar un método en el que se premia un reducido coste computacional y una alta efectividad en los resultados.

Finalmente, en cuanto a la legislación aplicable respecto al uso de robots en las plantas o fábricas industriales tenemos que hacer primero referencia al nivel de automatización europeo. Rumania es uno de los países con menor densidad de robots o proporción por trabajador, se estima que hay tan sólo once robots por cada diez mil empleados, lo cual se queda muy lejano de los aproximadamente setenta robots de media europea. Además, incluso los países vecinos como Eslovenia supera los cien robots por cada diez mil empleados.

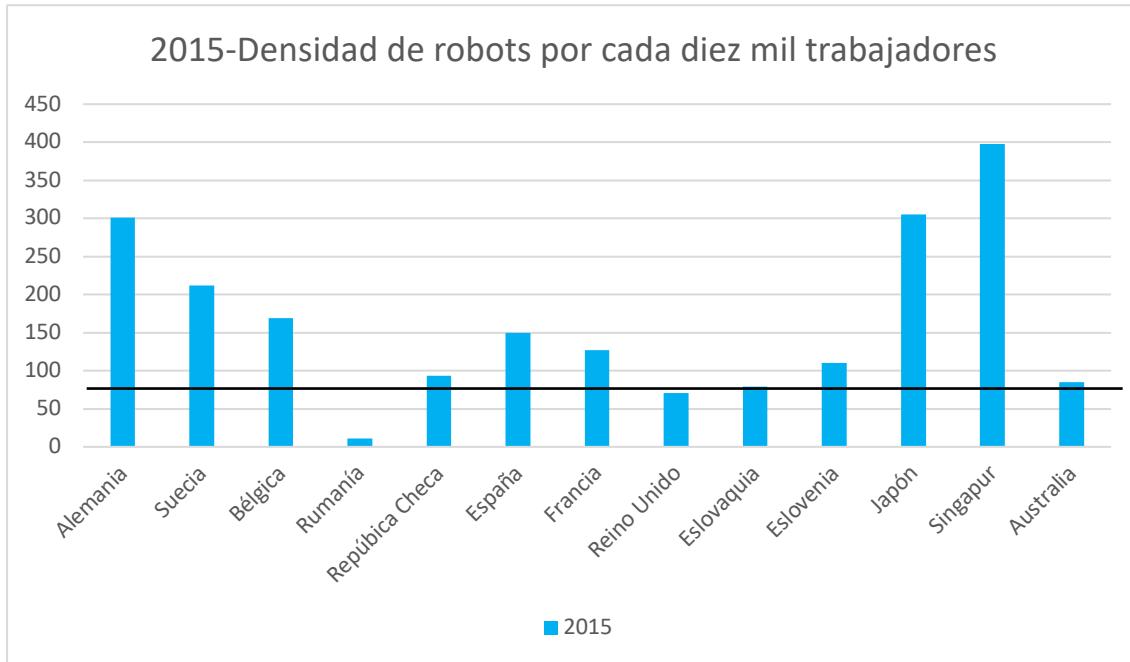


Fig. 83: Gráfica densidad de robots por cada diez mil trabajadores.

En Europa es de obligado cumplimiento la normativa CE que establece que se ha de dotar a toda el área de alcance de un robot industrial de un perímetro de seguridad para evitar que cualquier individuo pueda acceder a donde se encuentra el robot mientras éste está en funcionamiento. De esta forma, se utilizan la información respectiva al estado de la puerta del perímetro de seguridad (abierta o cerrada), al botón de emergencia (accionado o no) y el estado del robot (en marcha o parado). El recinto deberá contar con una alarma visual (rojo o verde) que indica al trabajador si puede entrar o no en el perímetro de seguridad y en caso de que éste entre cuando el robot esté en funcionamiento, se desactivará automáticamente la máquina.

Por otro lado, en las normas *ISO 10218-1* e *ISO 10218-2* se indican los requerimientos de seguridad para robots industriales en cuanto al uso, controlador, manipulador y respecto a las medidas de seguridad de la zona de alcance de la máquina tales como las explicadas anteriormente.

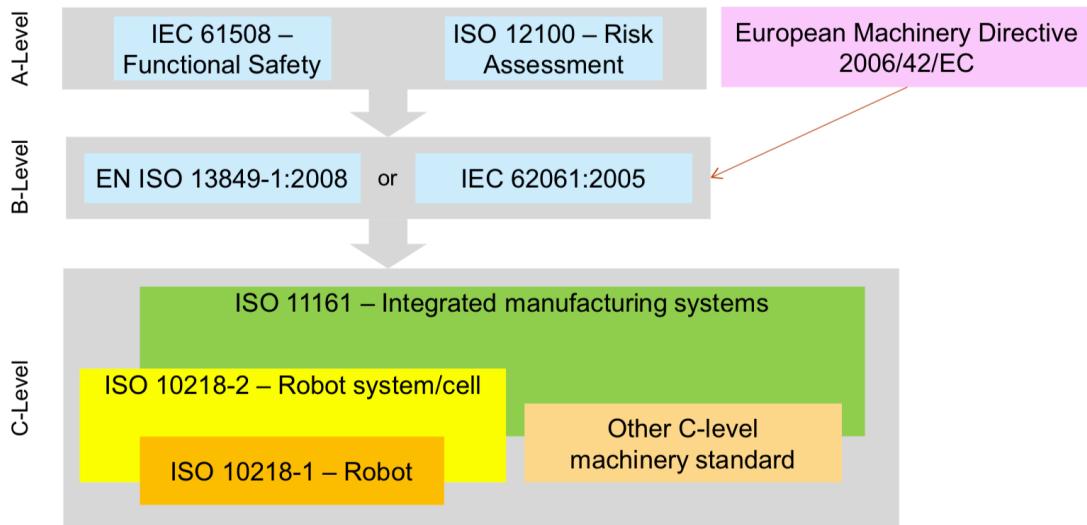


Fig. 84: Estándares de seguridad para la aplicación de los robots industriales. Imagen obtenida de Industrial Safety Requirements for Collaborative Robots and Applications de ABB.

8.3 PLANIFICACIÓN DEL PROYECTO

Para desarrollar el programa expuesto y como consecuente el Trabajo Fin de Grado, se ha planificado el proyecto de la siguiente manera.

Etapa I)

Profundizaje en el tema visión por computador y todos los pasos que conlleva.

Etapa II)

Análisis de las fotos de platos obtenidas y planteamiento de las diversas estrategias a seguir para resolver el problema planteado.

Etapa III)

Obtención de resultados a partir del programa creado y análisis de los mismos. Realización de mejoras.

ETAPA	FECHA INICIO	FECHA FIN	DÍAS
Investigación sobre el campo de visión por computador y MATLAB.	01/09/2017	01/10/2017	30
Planteamiento de estrategias y desarrollo del programa	08/10/2017	30/12/2017	86
Obtención y análisis de resultados.	13/01/2018	10/02/2018	28

Tabla 1: Planificación del proyecto.

8.4 PRESUPUESTO

A continuación, se dispone una tabla explicativa del presupuesto del proyecto.

PRESUPUESTO DEL PROYECTO																							
1. Realizado por:																							
Ángela Muñoz Moreno-Arrones																							
2. Objetivos del proyecto:																							
<u>Título:</u> Detección automática de defectos en artículos de cerámica. <u>Periodo de realización:</u> 2017/2018																							
3. Desglose de costes:																							
<table border="1"> <thead> <tr> <th>Descripción</th><th>Horas dedicadas</th><th>Coste</th><th>Amortización</th></tr> </thead> <tbody> <tr> <td>Ordenador</td><td></td><td>1.300 €</td><td>346,667</td></tr> <tr> <td>Licencia Matlab 2017b</td><td></td><td>69 €</td><td></td></tr> <tr> <td>Ángela Muñoz</td><td>1700</td><td>13.600 €</td><td></td></tr> <tr> <td>TOTAL</td><td>14.969 €</td><td></td><td>346,667</td></tr> </tbody> </table>				Descripción	Horas dedicadas	Coste	Amortización	Ordenador		1.300 €	346,667	Licencia Matlab 2017b		69 €		Ángela Muñoz	1700	13.600 €		TOTAL	14.969 €		346,667
Descripción	Horas dedicadas	Coste	Amortización																				
Ordenador		1.300 €	346,667																				
Licencia Matlab 2017b		69 €																					
Ángela Muñoz	1700	13.600 €																					
TOTAL	14.969 €		346,667																				
<table border="1"> <tbody> <tr> <td>Presupuesto total:</td><td></td><td></td><td></td></tr> <tr> <td>Amortización</td><td>346,667</td><td></td><td></td></tr> <tr> <td>Coste software y hardware</td><td>1.369 €</td><td></td><td></td></tr> <tr> <td>Personal</td><td>13600</td><td></td><td></td></tr> <tr> <td>Total</td><td>15315,667</td><td></td><td></td></tr> </tbody> </table>				Presupuesto total:				Amortización	346,667			Coste software y hardware	1.369 €			Personal	13600			Total	15315,667		
Presupuesto total:																							
Amortización	346,667																						
Coste software y hardware	1.369 €																						
Personal	13600																						
Total	15315,667																						
(Periodo de depreciación del ordenador: 30 meses)																							

8.5 CÓDIGO

8.5.1 PROGRAMA

```

function varargout = Programa(varargin)
% PROGRAMA MATLAB code for Programa.fig
%     PROGRAMA, by itself, creates a new PROGRAMA or raises the
existing
%     singleton*.
%
%     H = PROGRAMA returns the handle to a new PROGRAMA or the handle
to
%     the existing singleton*.
%
%     PROGRAMA( 'CALLBACK', hObject, eventData, handles,...) calls the
local
%     function named CALLBACK in PROGRAMA.M with the given input
arguments.
%
%     PROGRAMA('Property','Value',...) creates a new PROGRAMA or
raises the
%     existing singleton*. Starting from the left, property value
pairs are
%     applied to the GUI before Programa_OpeningFcn gets called. An
%     unrecognized property name or invalid value makes property
application
%     stop. All inputs are passed to Programa_OpeningFcn via
varargin.
%
%     *See GUI Options on GUIDE's Tools menu. Choose "GUI allows
only one
%     instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Programa

% Last Modified by GUIDE v2.5 02-Feb-2018 16:23:31

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',         mfilename, ...
                   'gui_Singleton',    gui_Singleton, ...
                   'gui_OpeningFcn',   @Programa_OpeningFcn, ...
                   'gui_OutputFcn',    @Programa_OutputFcn, ...
                   'gui_LayoutFcn',    [], ...
                   'gui_Callback',     []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before Programa is made visible.

```



```

function Programa_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata   reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Programa (see VARARGIN)

% Choose default command line output for Programa
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Programa wait for user response (see UIRESUME)
% uwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = Programa_OutputFcn(hObject, eventdata, handles)
% varargout cell array for returning output args (see VARARGOUT);
% hObject handle to figure
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes on selection change in listbox1.
function listbox1_Callback(hObject, eventdata, handles)
% hObject handle to listbox1 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns listbox1
contents as cell array
% contents{get(hObject,'Value')} returns selected item from
listbox1


% --- Executes during object creation, after setting all properties.
function listbox1_CreateFcn(hObject, eventdata, handles)
% hObject handle to listbox1 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles empty - handles not created until after all CreateFcns
called

% Hint: listbox controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in popupmenu1.
function popupmenu1_Callback(hObject, eventdata, handles)
% hObject handle to popupmenu1 (see GCBO)

```

```
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns popupmenu1
contents as cell array
%      contents{get(hObject,'Value')} returns selected item from
popupmenu1

str = get(hObject, 'String');
val = get(hObject, 'Value');
opcion=str(val);

switch cell2mat(opcion)

    case 'Tipo 1'

        im_detrás=imresize(rgb2gray(imread('Back1.png')),0.35);
        imop=imopen(im2bw(imadjust(im_detrás),0.35),strel('disk',10));
        imc=imclose(imop,strel('disk',9));
        im_detrás=im_detrás.*uint8(imc);
        im_detrás_logical=(im_detrás>20);
        detras_prop =
        regionprops('table',im_detrás_logical,'Centroid','MajorAxisLength','Mi
norAxisLength');
        center1=
        detras_prop.Centroid;MajorAxis1=detrás_prop.MajorAxisLength;MinorAxis1
=detrás_prop.MinorAxisLength;
        MajorAxis1h=ceil(MajorAxis1/2);MinorAxis1h=ceil(MinorAxis1/2);
        rect1=[round(center1(1))-MajorAxis1h-10,round(center1(2))-MinorAxis1h-
10,MajorAxis1+20,MinorAxis1+20];
        im_detrás_crop=imcrop(im_detrás,rect1);
        Low_High=stretchlim(im_detrás_crop);
        D=imadjust(im_detrás_crop,Low_High);
        [imgCOR,threshOut] = edge(D,'LOG',0.009,2.42);

        im_d=imresize(rgb2gray(imread('DB1.png')),0.37);
        imop=imopen(im2bw(imadjust(im_d),0.35),strel('disk',10));
        imc=imclose(imop,strel('disk',9));
        im_DB=im_d.*uint8(imc);
        im_DB_logical=(im_DB>20);
        DB_prop =
        regionprops('table',im_DB_logical,'Centroid','MajorAxisLength','Minora
xisLength');
        center2=
        DB_prop.Centroid;MajorAxis2=DB_prop.MajorAxisLength;MinorAxis2=DB_prop
.MinorAxisLength;
        c1_x=center1(1,1);c2_x=center2(1,1);c1_y=center1(1,2);c2_y=center2(1,2
);
        im_DB_crop=imtranslate(im_DB,[c1_x-c2_x,c1_y-c2_y-10]);
        im_DB_crop=imcrop(im_DB_crop,rect1);
        im_DB_crop=imresize(im_DB_crop,size(im_detrás_crop));
        Low_High2=stretchlim(im_DB_crop);
        DC=imadjust(im_DB_crop,Low_High2);
        [imgDEF,threshOut] = edge(DC,'LOG',0.003,3.47);
        Segout2=Funcion(D,imgDEF,imgCOR);
        imshow(Segout2,'Parent',handles.axes4);
        imshow(D,'Parent',handles.axes3);

        im_detrás=imresize(rgb2gray(imread('Front1.png')),0.1908);
        imop=imopen(im2bw(imadjust(im_detrás),0.35),strel('disk',10));
```

```

imc=imclose(imop,strel('disk',9));
im_detrás=im_detrás.*uint8(imc);
im_detrás_logical=(im_detrás>20);
detrás_prop =
regionprops('table',im_detrás_logical,'Centroid','MajorAxisLength','Mi
norAxisLength');
center1=
detrás_prop.Centroid;MajorAxis1=detrás_prop.MajorAxisLength;MinorAxis1
=detrás_prop.MinorAxisLength;
MajorAxis1h=ceil(MajorAxis1/2);MinorAxis1h=ceil(MinorAxis1/2);
rect1=[round(center1(1))-MajorAxis1h-10,round(center1(2))-MinorAxis1h-
10,MajorAxis1+20,MinorAxis1+20];
im_detrás_crop=imcrop(im_detrás,rect1);
Low_High=stretchlim(im_detrás_crop);
D=imadjust(im_detrás_crop,Low_High);
[imgCOR threshOut] = edge(D,'LOG',0.009,2.42);

im_d=imresize(rgb2gray(imread('DF1.png')),0.2);
imop=imopen(im2bw(imadjust(im_d),0.35),strel('disk',10));
imc=imclose(imop,strel('disk',9));
im_DB=im_d.*uint8(imc);
im_DB_logical=(im_DB>20);
DB_prop =
regionprops('table',im_DB_logical,'Centroid','MajorAxisLength','MinorA
xisLength');
center2=
DB_prop.Centroid;MajorAxis2=DB_prop.MajorAxisLength;MinorAxis2=DB_prop
.MinorAxisLength;
c1_x=center1(1,1);c2_x=center2(1,1);c1_y=center1(1,2);c2_y=center2(1,2
);
im_DB_crop=imtranslate(im_DB,[c1_x-c2_x,c1_y-c2_y-6]);
im_DB_crop=imcrop(im_DB_crop,rect1);
im_DB_crop=imresize(im_DB_crop,size(im_detrás_crop));
Low_High2=stretchlim(im_DB_crop);
DC=imadjust(im_DB_crop,Low_High2);
[imgDEF threshOut] = edge(DC,'LOG',0.003,3.47);
Segout2=Funcion(D,imgDEF,imgCOR);
    imshow(Segout2,'Parent',handles.axes2);
    imshow(D,'Parent',handles.axes1);
set(handles.text2, 'String', 'Plato roto')

case 'Tipo 2'

    im_detrás=imresize(rgb2gray(imread('Back2.png')),0.35);
imop=imopen(im2bw(imadjust(im_detrás),0.35),strel('disk',10));
imc=imclose(imop,strel('disk',9));
im_detrás=im_detrás.*uint8(imc);
im_detrás_logical=(im_detrás>20);
detrás_prop =
regionprops('table',im_detrás_logical,'Centroid','MajorAxisLength','Mi
norAxisLength');
center1=
detrás_prop.Centroid;MajorAxis1=detrás_prop.MajorAxisLength;MinorAxis1
=detrás_prop.MinorAxisLength;
MajorAxis1h=ceil(MajorAxis1/2);MinorAxis1h=ceil(MinorAxis1/2);
rect1=[round(center1(1))-MajorAxis1h-10,round(center1(2))-MinorAxis1h-
10,MajorAxis1+20,MinorAxis1+20];
im_detrás_crop=imcrop(im_detrás,rect1);
Low_High=stretchlim(im_detrás_crop);
D=imadjust(im_detrás_crop,Low_High);
[imgCOR threshOut] = edge(D,'LOG',0.003927,2.00);

```

```

im_d=imresize(rgb2gray(imread('DB2.png')),0.35);
imop=imopen(im2bw(imadjust(im_d),0.35),strel('disk',10));
imc=imclose(imop,strel('disk',9));
im_DB=im_d.*uint8(imc);
im_DB_logical=(im_DB>20);
DB_prop =
regionprops('table',im_DB_logical,'Centroid','MajorAxisLength','MinorA
xisLength');
center2=
DB_prop.Centroid;MajorAxis2=DB_prop.MajorAxisLength;MinorAxis2=DB_prop
.MinorAxisLength;
c1_x=center1(1,1);c2_x=center2(1,1);c1_y=center1(1,2);c2_y=center2(1,2
);
im_DB_crop=imtranslate(im_DB,[c1_x-c2_x,c1_y-c2_y]);
im_DB_crop=imcrop(im_DB_crop,rect1);
im_DB_crop=imresize(im_DB_crop,size(im_detrás_crop));
Low_High2=stretchlim(im_DB_crop);
DC=imadjust(im_DB_crop,Low_High2);
[imgDEF,threshOut] = edge(DC,'LOG',0.002833,1.93); %2.0
Segout2=Funcion(D,imgDEF,imgCOR);
    imshow(Segout2,'Parent',handles.axes4);
    imshow(D,'Parent',handles.axes3);

im_detrás=imresize(rgb2gray(imread('Front2.png')),0.35);
imop=imopen(im2bw(imadjust(im_detrás),0.35),strel('disk',10));
imc=imclose(imop,strel('disk',9));
im_detrás=im_detrás.*uint8(imc);
im_detrás_logical=(im_detrás>20);
detrás_prop =
regionprops('table',im_detrás_logical,'Centroid','MajorAxisLength','Mi
norAxisLength');
center1=
detrás_prop.Centroid;MajorAxis1=detrás_prop.MajorAxisLength;MinorAxis1
=detrás_prop.MinorAxisLength;
MajorAxis1h=ceil(MajorAxis1/2);MinorAxis1h=ceil(MinorAxis1/2);
rect1=[round(center1(1))-MajorAxis1h-10,round(center1(2))-MinorAxis1h-
10,MajorAxis1+20,MinorAxis1+20];
im_detrás_crop=imcrop(im_detrás,rect1);
Low_High=stretchlim(im_detrás_crop);
D=imadjust(im_detrás_crop,Low_High);
[imgCOR,threshOut] = edge(D,'LOG',0.003927,2.00);

im_d=imresize(rgb2gray(imread('DF2.png')),0.3483);
imop=imopen(im2bw(imadjust(im_d),0.35),strel('disk',10));
imc=imclose(imop,strel('disk',9));
im_DB=im_d.*uint8(imc);
im_DB_logical=(im_DB>20);
DB_prop =
regionprops('table',im_DB_logical,'Centroid','MajorAxisLength','MinorA
xisLength');
center2=
DB_prop.Centroid;MajorAxis2=DB_prop.MajorAxisLength;MinorAxis2=DB_prop
.MinorAxisLength;
c1_x=center1(1,1);c2_x=center2(1,1);c1_y=center1(1,2);c2_y=center2(1,2
);
im_DB_crop=imtranslate(im_DB,[c1_x-c2_x,c1_y-c2_y]);
im_DB_crop=imcrop(im_DB_crop,rect1);
im_DB_crop=imresize(im_DB_crop,size(im_detrás_crop));
Low_High2=stretchlim(im_DB_crop);
DC=imadjust(im_DB_crop,Low_High2);
[imgDEF,threshOut] = edge(DC,'LOG',0.002,3.33);

```

```

Segout2=Funcion(D,imgDEF,imgCOR);
    imshow(Segout2,'Parent',handles.axes2);
    imshow(D,'Parent',handles.axes1);
set(handles.text2, 'String', 'Rotura en el borde interior delantero')

case 'Tipo 3'

    im_detrás=imresize(rgb2gray(imread('Front3.png')),0.35);
imop=imopen(im2bw(imadjust(im_detrás),0.35),strel('disk',10));
imc=imclose(imop,strel('disk',9));
im_detrás=im_detrás.*uint8(imc);
im_detrás_logical=(im_detrás>20);
detrás_prop =
regionprops('table',im_detrás_logical,'Centroid','MajorAxisLength','Mi
norAxisLength');
center1=
detrás_prop.Centroid;MajorAxis1=detrás_prop.MajorAxisLength;MinorAxis1
=detrás_prop.MinorAxisLength;
MajorAxis1h=ceil(MajorAxis1/2);MinorAxis1h=ceil(MinorAxis1/2);
rect1=[round(center1(1))-MajorAxis1h-10,round(center1(2))-MinorAxis1h-
10,MajorAxis1+20,MinorAxis1+20];
im_detrás_crop=imcrop(im_detrás,rect1);
Low_High=stretchlim(im_detrás_crop);
D=imadjust(im_detrás_crop,Low_High);
[imgCOR,threshOut] = edge(D,'LOG',0.004,1.72);

im_d=imresize(rgb2gray(imread('DF3.png')),0.35);
imop=imopen(im2bw(imadjust(im_d),0.35),strel('disk',10));
imc=imclose(imop,strel('disk',9));
im_DB=im_d.*uint8(imc);
im_DB_logical=(im_DB>20);
DB_prop =
regionprops('table',im_DB_logical,'Centroid','MajorAxisLength','Minora
xisLength');
center2=
DB_prop.Centroid;MajorAxis2=DB_prop.MajorAxisLength;MinorAxis2=DB_prop
.MinorAxisLength;
c1_x=center1(1,1);c2_x=center2(1,1);c1_y=center1(1,2);c2_y=center2(1,2
);
im_DB_crop=imtranslate(im_DB,[c1_x-c2_x,c1_y-c2_y]);
im_DB_crop=imcrop(im_DB_crop,rect1);
im_DB_crop=imresize(im_DB_crop,size(im_detrás_crop));
Low_High2=stretchlim(im_DB_crop);
DC=imadjust(im_DB_crop,Low_High2);
[imgDEF,threshOut] = edge(DC,'LOG',0.001,1.8);
Segout2=Funcion(D,imgDEF,imgCOR);
    imshow(Segout2,'Parent',handles.axes2);
    imshow(D,'Parent',handles.axes1);

im_detrás=imresize(rgb2gray(imread('Back3.png')),0.5);
imop=imopen(im2bw(imadjust(im_detrás),0.35),strel('disk',10));
imc=imclose(imop,strel('disk',9));
im_detrás=im_detrás.*uint8(imc);
im_detrás_logical=(im_detrás>20);
detrás_prop =
regionprops('table',im_detrás_logical,'Centroid','MajorAxisLength','Mi
norAxisLength');
center1=
detrás_prop.Centroid;MajorAxis1=detrás_prop.MajorAxisLength;MinorAxis1
=detrás_prop.MinorAxisLength;

```

```

MajorAxis1h=ceil(MajorAxis1/2);MinorAxis1h=ceil(MinorAxis1/2);
rect1=[round(center1(1))-MajorAxis1h-10,round(center1(2))-MinorAxis1h-
10,MajorAxis1+20,MinorAxis1+20];
im_detrás_crop=imcrop(im_detrás,rect1);
Low_High=stretchlim(im_detrás_crop);
D=imadjust(im_detrás_crop,Low_High);
[imgCOR,threshOut] = edge(D,'LOG',0.015,1.58);

im_d=imresize(rgb2gray(imread('DB3.png')),0.5);
imop=imopen(im2bw(imadjust(im_d),0.35),strel('disk',10));
imc=imclose(imop,strel('disk',9));
im_DB=im_d.*uint8(imc);
im_DB_logical=(im_DB>20);
DB_prop =
regionprops('table',im_DB_logical,'Centroid','MajorAxisLength','MinorA
xisLength');
center2=
DB_prop.Centroid;MajorAxis2=DB_prop.MajorAxisLength;MinorAxis2=DB_prop
.MinorAxisLength;
c1_x=center1(1,1);c2_x=center2(1,1);c1_y=center1(1,2);c2_y=center2(1,2
);
im_DB_crop=imtranslate(im_DB,[c1_x-c2_x,c1_y-c2_y]);
im_DB_crop=imcrop(im_DB_crop,rect1);
im_DB_crop=imresize(im_DB_crop,size(im_detrás_crop));
Low_High2=stretchlim(im_DB_crop);
DC=imadjust(im_DB_crop,Low_High2);
[imgDEF,threshOut] = edge(DC,'LOG',0.18,0.55);
Segout2=Funcion(D,imgDEF,imgCOR);
    imshow(Segout2,'Parent',handles.axes4);
    imshow(D,'Parent',handles.axes3);
set(handles.text2, 'String', 'Plato cascarillado por delante y
detrás')

case 'Tipo 4'

    im_detrás=imresize(rgb2gray(imread('Front4.png')),0.35);
imop=imopen(im2bw(imadjust(im_detrás),0.35),strel('disk',10));
imc=imclose(imop,strel('disk',9));
im_detrás=im_detrás.*uint8(imc);
im_detrás_logical=(im_detrás>20);
detrás_prop =
regionprops('table',im_detrás_logical,'Centroid','MajorAxisLength','Mi
norAxisLength');
center1=
detrás_prop.Centroid;MajorAxis1=detrás_prop.MajorAxisLength;MinorAxis1
=detrás_prop.MinorAxisLength;
MajorAxis1h=ceil(MajorAxis1/2);MinorAxis1h=ceil(MinorAxis1/2);
rect1=[round(center1(1))-MajorAxis1h-10,round(center1(2))-MinorAxis1h-
10,MajorAxis1+20,MinorAxis1+20];
im_detrás_crop=imcrop(im_detrás,rect1);
Low_High=stretchlim(im_detrás_crop);
D=imadjust(im_detrás_crop,Low_High);
[imgCOR,threshOut] = edge(D,'LOG',0.074697,0.62);

im_d=imresize(rgb2gray(imread('DF4.png')),0.374);
imop=imopen(im2bw(imadjust(im_d),0.35),strel('disk',10));
imc=imclose(imop,strel('disk',9));
im_DB=im_d.*uint8(imc);
im_DB_logical=(im_DB>20);
DB_prop =
regionprops('table',im_DB_logical,'Centroid','MajorAxisLength','MinorA
xisLength');

```

```

center2=
DB_prop.Centroid;MajorAxis2=DB_prop.MajorAxisLength;MinorAxis2=DB_prop
.MinorAxisLength;
c1_x=center1(1,1);c2_x=center2(1,1);c1_y=center1(1,2);c2_y=center2(1,2
);
im_DB_crop=imtranslate(im_DB,[c1_x-c2_x,c1_y-c2_y]);
im_DB_crop=imcrop(im_DB_crop,rect1);
im_DB_crop=imresize(im_DB_crop,size(im_detrás_crop));
Low_High2=stretchlim(im_DB_crop);
DC=imadjust(im_DB_crop,Low_High2);
[imgDEF,threshOut] = edge(DC,'LOG',0.082,0.58);
Segout2=Funcion(D,imgDEF,imgCOR);
    imshow(Segout2,'Parent',handles.axes2);
    imshow(D,'Parent',handles.axes1);

im_detrás=imresize(rgb2gray(imread('Back4.png')),0.35);
imop=imopen(im2bw(imadjust(im_detrás),0.35),strel('disk',10));
imc=imclose(imop,strel('disk',9));
im_detrás=im_detrás.*uint8(imc);
im_detrás_logical=(im_detrás>20);
detrás_prop =
regionprops('table',im_detrás_logical,'Centroid','MajorAxisLength','Mi
norAxisLength');
center1=
detrás_prop.Centroid;MajorAxis1=detrás_prop.MajorAxisLength;MinorAxis1
=detrás_prop.MinorAxisLength;
MajorAxis1h=ceil(MajorAxis1/2);MinorAxis1h=ceil(MinorAxis1/2);
rect1=[round(center1(1))-MajorAxis1h-10,round(center1(2))-MinorAxis1h-
10,MajorAxis1+20,MinorAxis1+20];
im_detrás_crop=imcrop(im_detrás,rect1);
Low_High=stretchlim(im_detrás_crop);
D=imadjust(im_detrás_crop,Low_High);
[imgCOR,threshOut] = edge(D,'LOG',0.015218,2.00);

im_d=imresize(rgb2gray(imread('DB4.png')),0.38);
imop=imopen(im2bw(imadjust(im_d),0.35),strel('disk',10));
imc=imclose(imop,strel('disk',9));
im_DB=im_d.*uint8(imc);
im_DB_logical=(im_DB>20);
DB_prop =
regionprops('table',im_DB_logical,'Centroid','MajorAxisLength','Minora
xisLength');
center2=
DB_prop.Centroid;MajorAxis2=DB_prop.MajorAxisLength;MinorAxis2=DB_prop
.MinorAxisLength;
c1_x=center1(1,1);c2_x=center2(1,1);c1_y=center1(1,2);c2_y=center2(1,2
);
im_DB_crop=imtranslate(im_DB,[c1_x-c2_x,c1_y-c2_y]);
im_DB_crop=imcrop(im_DB_crop,rect1);
im_DB_crop=imresize(im_DB_crop,size(im_detrás_crop));
Low_High2=stretchlim(im_DB_crop);
DC=imadjust(im_DB_crop,Low_High2);
[imgDEF,threshOut] = edge(DC,'LOG',0.003,1.65);
Segout2=Funcion(D,imgDEF,imgCOR);
    imshow(Segout2,'Parent',handles.axes4);
    imshow(D,'Parent',handles.axes3);
set(handles.text2, 'String', 'Rotura en el borde interior trasero')

case 'Tipo 5'

    im_detrás=imresize(rgb2gray(imread('Front5.png')),0.35);
imop=imopen(im2bw(imadjust(im_detrás),0.35),strel('disk',10));

```



```

imc=imclose(imop,strel('disk',9));
im_detrás=im_detrás.*uint8(imc);
im_detrás_logical=(im_detrás>20);
detrás_prop =
regionprops('table',im_detrás_logical,'Centroid','MajorAxisLength','Mi
norAxisLength');
center1=
detrás_prop.Centroid;MajorAxis1=detrás_prop.MajorAxisLength;MinorAxis1
=detrás_prop.MinorAxisLength;
MajorAxis1h=ceil(MajorAxis1/2);MinorAxis1h=ceil(MinorAxis1/2);
rect1=[round(center1(1))-MajorAxis1h-10,round(center1(2))-MinorAxis1h-
10,MajorAxis1+20,MinorAxis1+20];
im_detrás_crop=imcrop(im_detrás,rect1);
Low_High=stretchlim(im_detrás_crop);
D=imadjust(im_detrás_crop,Low_High);
[imgCOR,threshOut] = edge(D,'LOG',0.011,1.86);

im_d=imresize(rgb2gray(imread('DF5.png')),0.35);
imop=imopen(im2bw(imadjust(im_d),0.35),strel('disk',10));
imc=imclose(imop,strel('disk',9));
im_DB=im_d.*uint8(imc);
im_DB_logical=(im_DB>20);
DB_prop =
regionprops('table',im_DB_logical,'Centroid','MajorAxisLength','MinorA
xisLength');
center2=
DB_prop.Centroid;MajorAxis2=DB_prop.MajorAxisLength;MinorAxis2=DB_prop
.MinorAxisLength;
c1_x=center1(1,1);c2_x=center2(1,1);c1_y=center1(1,2);c2_y=center2(1,2
);
im_DB_crop=imtranslate(im_DB,[c1_x-c2_x,c1_y-c2_y]);
im_DB_crop=imcrop(im_DB_crop,rect1);
im_DB_crop=imresize(im_DB_crop,size(im_detrás_crop));
Low_High2=stretchlim(im_DB_crop);
DC=imadjust(im_DB_crop,Low_High2);
[imgDEF,threshOut] = edge(DC,'LOG',0.017,1.86);
Segout2=Funcion(D,imgDEF,imgCOR);
imshow(Segout2,'Parent',handles.axes2);
imshow(D,'Parent',handles.axes1);

im_detrás=imresize(rgb2gray(imread('Back5.png')),0.35);
imop=imopen(im2bw(imadjust(im_detrás),0.35),strel('disk',10));
imc=imclose(imop,strel('disk',9));
im_detrás=im_detrás.*uint8(imc);
im_detrás_logical=(im_detrás>20);
detrás_prop =
regionprops('table',im_detrás_logical,'Centroid','MajorAxisLength','Mi
norAxisLength');
center1=
detrás_prop.Centroid;MajorAxis1=detrás_prop.MajorAxisLength;MinorAxis1
=detrás_prop.MinorAxisLength;
MajorAxis1h=ceil(MajorAxis1/2);MinorAxis1h=ceil(MinorAxis1/2);
rect1=[round(center1(1))-MajorAxis1h-10,round(center1(2))-MinorAxis1h-
10,MajorAxis1+20,MinorAxis1+20];
im_detrás_crop=imcrop(im_detrás,rect1);
Low_High=stretchlim(im_detrás_crop);
D=imadjust(im_detrás_crop,Low_High);
[imgCOR,threshOut] = edge(D,'LOG',0.017,1.86);

im_d=imresize(rgb2gray(imread('DB5.png')),0.35);
imop=imopen(im2bw(imadjust(im_d),0.35),strel('disk',10));
imc=imclose(imop,strel('disk',9));

```

```

im_DB=im_d.*uint8(imc);
im_DB_logical=(im_DB>20);
DB_prop =
regionprops('table',im_DB_logical,'Centroid','MajorAxisLength','MinorA
xisLength');
center2=
DB_prop.Centroid;MajorAxis2=DB_prop.MajorAxisLength;MinorAxis2=DB_prop
.MinorAxisLength;
c1_x=center1(1,1);c2_x=center2(1,1);c1_y=center1(1,2);c2_y=center2(1,2
);
im_DB_crop=imtranslate(im_DB,[c1_x-c2_x,c1_y-c2_y]);
im_DB_crop=imcrop(im_DB_crop,rect1);
im_DB_crop=imresize(im_DB_crop,size(im_detrás_crop));
Low_High2=stretchlim(im_DB_crop);
DC=imadjust(im_DB_crop,Low_High2);
[imgDEF,threshOut] = edge(DC,'LOG',0.017,1.86);
Segout2=Funcion(D,imgDEF,imgCOR);
    imshow(Segout2,'Parent',handles.axes4);
    imshow(D,'Parent',handles.axes3);
set(handles.text2, 'String', 'Plato correcto')

end

% --- Executes during object creation, after setting all properties.
function popupmenu1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu1 (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

```

8.5.2 FUNCIÓN OBTENER DEFECTO

```

function [Segout2]=Funcion(D,imgDEF,imgCOR)

SE1 = strel('Disk',10,4);
morphed1 = imclose(imgDEF, SE1);
SE2 = strel('Disk',1,4);
morphed2 = imtophat(morphed1, SE2);
resultado=imabsdiff(morphed2,imgDEF);
resultado=bwareaopen(resultado,10);
[r,c]=size(resultado);

contador=0;
for col=1:c
    for fila=1:r
        if resultado(fila,col)==1
            contador=1;
        end
    end
end
if contador==1

```

DETECCIÓN AUTOMÁTICA DE DEFECTOS EN ARTÍCULOS DE CERÁMICA



```
Segout2=imoverlay(D, resultado, [1 0 0]);  
  
else  
J=imabsdiff(imgDEF,imgCOR);  
img2=bwareaopen(J,5);  
Segout2=imoverlay(D, img2, [1 0 0]);  
  
end
```



9. ÍNDICE DE FIGURAS

FIG. 1: EJEMPLO VISIÓN POR COMPUTADOR	9
FIG. 2: FUNCIONES DE PERTENENCIA DE DOS CONJUNTOS. FUENTE DIGITAL IMAGE PROCESSING DE GONZÁLEZ	12
FIG. 3: PLATO PARTE DELANTERA A DETECTAR DEFECTOS.....	16
FIG. 4: RESULTADO IMAGEN AL APLICAR LA TRANSFORMADA DE HOUGH	16
FIG. 5: DEFECTO DETECTADO TRAS APLICAR EL FILTRO LOG A LA IMAGEN	17
FIG. 6: DETECCIÓN DEFECTO DEL PLATO CON RESULTADOS ERRÓNEOS	18
FIG. 7: PLATO CON DEFECTO EN LA PARTE TRASERA Y BORDE IRREGULAR	18
FIG. 8: DETECCIÓN DE BORDES FALLIDA CON EL NÚMERO DE EULER.....	19
FIG. 9: DETECCIÓN DE BORDES ZEROCROSS SOBEL.....	19
FIG. 10: DEFECTO MARCADO IMAGEN BINARIA	20
FIG. 11: DEFECTO MARCADO IMAGEN PLATO	20
FIG. 12: MONTAJE IMAGEN ORIGINAL Y FILTRADA MEDIANTE EL FILTRO LAPLACIANO LOCAL Y RÁPIDO.	21
FIG. 13: IMAGEN FRONTAL PLATO BORDES DETECTADOS	21
FIG. 14: RESULTADO DETECCIÓN DE DEFECTOS FILTRO LAPLACIANO LOCAL Y RÁPIDO	22
FIG. 15: RESULTADO DETECCIÓN DE DEFECTOS MEDIANTE FUZZY	22
FIG. 16: RESULTADO DETECCIÓN DE DEFECTOS MEDIANTE FUZZY CON NUEVAS FUNCIONES DE PERTENENCIA	23
FIG. 17: PLATO CON ROTURA EN EL BORDE INTERNO	24
FIG. 18: RESULTADO FILTRO ROBERTS	24
FIG. 19: RESULTADO FILTRO PREWITT.....	25
FIG. 20: RESULTADO FILTRO SOBEL	25
FIG. 21: IMAGEN PLATO PARTE TRASERA A FILTRAR.....	26
FIG. 22: IMAGEN PLATO PARTE TRASERA TRAS APLICAR CLOSING	26
FIG. 23: IMAGEN PLATO PARTE TRASERA TRAS APLICAR OPENING	27
FIG. 24: RESULTADO DETECCIÓN DE DEFECTOS SOBEL Y OPERACIONES MORFOLÓGICAS.....	27
FIG. 25: RESULTADO FILTRO CANNY	28
FIG. 26: RESULTADO FILTRO LOG.....	28
FIG. 27: INTERFAZ PROGRAMA DETECCIÓN DE DEFECTOS EN ARTÍCULOS DE CERÁMICA	31
FIG. 28: DIAGRAMA DE FLUJO DEL PROCESO DE DETECCIÓN DE DEFECTOS EN LA FÁBRICA	32
FIG. 29: DIAGRAMA DE FLUJO ANÁLISIS IMAGEN.....	34
FIG. 30: FRONTAL PLATO CORRECTO TIPO 1	35
FIG. 31: TRASERA PLATO CORRECTO TIPO 1	35
FIG. 32: FOTO FRONTAL PARA ANALIZAR TIPO 1.....	36
FIG. 33: FOTO TRASERA PARA ANALIZAR TIPO 1.....	36
FIG. 34: FOTO FRONTAL PARA ANALIZAR BINARIA TIPO 1	37
FIG. 35: FOTO TRASERA PARA ANALIZAR BINARIA TIPO 1	37
FIG. 36: FOTO FRONTAL PARA ANALIZAR FILTRADA TIPO 1.....	38
FIG. 37: FOTO TRASERA PARA ANALIZAR FILTRADA TIPO 1	38
FIG. 38: IMAGEN FRONTAL RESULTADO TIPO 1	39
FIG. 39: IMAGEN TRASERA RESULTADO TIPO 1	39
FIG. 40: FOTO FRONTAL PLATO CORRECTO TIPO 2	40
FIG. 41: FOTO TRASERA PLATO CORRECTO TIPO 2	40
FIG. 42: FOTO FRONTAL PARA ANALIZAR TIPO 2.....	41
FIG. 43: FOTO TRASERA PARA ANALIZAR TIPO 2	41
FIG. 44: FOTO FRONTAL PARA ANALIZAR BINARIA TIPO 2	42



FIG. 45: FOTO TRASERA PARA ANALIZAR BINARIA TIPO 2	42
FIG. 46: FOTO FRONTEL FILTRADA PARA ANALIZAR TIPO 2	43
FIG. 47: FOTO TRASERA FILTRADA PARA ANALIZAR TIPO 2	43
FIG. 48: IMAGEN FRONTEL RESULTADO TIPO 2	44
FIG. 49: IMAGEN TRASERA RESULTADO TIPO 2	44
FIG. 50: FOTO FRONTEL PLATO CORRECTO TIPO 3	45
FIG. 51: FOTO TRASERA PLATO CORRECTO TIPO 3	45
FIG. 52: FOTO FRONTEL PARA ANALIZAR TIPO 3	46
FIG. 53: FOTO TRASERA PARA ANALIZAR TIPO 3	46
FIG. 54: FOTO FRONTEL PARA ANALIZAR BINARIA TIPO 3	47
FIG. 55: FOTO TRASERA PARA ANALIZAR BINARIA TIPO 3	47
FIG. 56: FOTO FRONTEL PARA ANALIZAR BINARIA TIPO 3	48
FIG. 57: FOTO TRASERA PARA ANALIZAR BINARIA TIPO 3	48
FIG. 58: IMAGEN FRONTEL RESULTADO TIPO 3	49
FIG. 59: IMAGEN TRASERA RESULTADO TIPO 3	49
FIG. 60: FOTO FRONTEL PLATO CORRECTO TIPO 4	50
FIG. 61: FOTO TRASERA PLATO CORRECTO TIPO 4	50
FIG. 62: FOTO FRONTEL PARA ANALIZAR TIPO 4	51
FIG. 63: FOTO TRASERA PARA ANALIZAR TIPO 4	51
FIG. 64: FOTO FRONTEL PARA ANALIZAR BINARIA TIPO 4	52
FIG. 65: FOTO TRASERA PARA ANALIZAR BINARIA TIPO 4	52
FIG. 66: FOTO FRONTEL PARA ANALIZAR FILTRADA TIPO 4	53
FIG. 67: FOTO TRASERA PARA ANALIZAR FILTRADA TIPO 4	53
FIG. 68: IMAGEN FRONTEL RESULTADO TIPO 4	54
FIG. 69: IMAGEN TRASERA RESULTADO TIPO 4	54
FIG. 70: FOTO FRONTEL PLATO CORRECTO TIPO 5	55
FIG. 71: FOTO TRASERA PLATO CORRECTO TIPO 5	55
FIG. 72: FOTO FRONTEL PARA ANALIZAR TIPO 5	56
FIG. 73: FOTO TRASERA PARA ANALIZAR TIPO 5	56
FIG. 74: FOTO FRONTEL ANALIZAR BINARIA TIPO 5	57
FIG. 75: FOTO TRASERA PARA ANALIZAR BINARIA TIPO 5	57
FIG. 76: FOTO FRONTEL PARA ANALIZAR FILTRADA TIPO 5	58
FIG. 77: FOTO TRASERA PARA ANALIZAR BINARIA TIPO 5	58
FIG. 78: IMAGEN RESULTADO FRONTEL TIPO 5	59
FIG. 79: IMAGEN RESULTADO TRASERO TIPO 5	59
FIG. 80: EJEMPLO DE DETECCIÓN MEDIANTE DEEP LEARNING	61
FIG. 81: FÁBRICA IPEC	62
FIG. 82: FÁBRICA IPEC DURANTE REVISIÓN MANUAL DE DEFECTOS	63
FIG. 83: GRÁFICA DENSIDAD DE ROBOTS POR CADA DIEZ MIL TRABAJADORES	64
FIG. 84: ESTÁNDARES DE SEGURIDAD PARA LA APLICACIÓN DE LOS ROBOTS INDUSTRIALES. IMAGEN OBTENIDA DE INDUSTRIAL SAFETY REQUIREMENTS FOR COLLABORATIVE ROBOTS AND APPLICATIONS DE ABB	65



10. ÍNDICE DE TABLAS

TABLA 1: PLANIFICACIÓN DEL PROYECTO	65
---	----



11. BIBLIOGRAFÍA

- Woods, R. E. & Gonzalez, R. C., 2013. *Digital Image Processing*. 3^a edición ed. Inglaterra: Pearson Addison-Wesley.
- Escalera Hueso, A. d. l., 2001. *Visión por computador. Fundamentos y métodos..* 1^a edición ed. España: Prentice Hall.
- Mathworks, (2017). *Matlab Documentation*. [En línea]
Available at:
https://es.mathworks.com/help/vision/index.html?searchHighlight=computer%20vision&s_tid=doc_srcTitle
- Qidway, U. & Chen, C., 2009. *Digital Image processing*. 1^a edición ed. Chapman & Hall/CRC Textbooks in Computing.
- Russ, J. C. & Neal, F., 2015. *The Image Processing Handbook*. 7^a edición ed. CRC Press.
- Solomon, C. & Breckon, T. P., 2010. *Fundamentals of Digital Image Processing: A Practical Approach with Examples in Matlab..* 1^a edición ed. Wiley-Blackwell.
- Yaroslavsky, L. P., July 2014. *Advanced Digital Imaging Laboratory Using MATLAB*.
- Zhou, H., Wu, J. & Zhang, J., 2010. *Digital Image Processing*.
- MATLAB and Computer Vision Toolbox Release 2017b, The MathWorks, Inc., Natick, Massachusetts, United States.
- MATLAB and Segmentation Toolbox Release 2012b, The MathWorks, Inc., Natick, Massachusetts, United States.