Proyecto:

Clasificación y localización de defectos en superficies de metal.

Especificación de Requisitos Software (ERS)

Autores:
Natalia Lourdes Pérez García de la Puente
Vicente Gilabert Maño
Luis Rosario Tremoulet
Modelo: IEEE/ANSI 830-1998

Contenidos

1 Introducción

Este documento es una Especificación de Requisitos Software (ERS) para el sistema de detección de defectos en imágenes de superficies de metal. Esta especificación se ha estructurado basándose en las directrices dadas por el estándar IEEE Práctica Recomendada para Especificaciones de Requisitos Software ANSI/IEEE 830, 1998.

El repositorio del proyecto donde se va a desarrollar el sistema es:

https://github.com/Luisrosario2604/AIVA_2022_METAL

1.1 Propósito del documento

El presente documento tiene como propósito la definición de las especificaciones para el desarrollo de un sistema de visión artificial para la clasificación y localización de defectos en superficies metálicas. Este producto será puesto en marcha en la empresa METALIKA. SL, cuyo responsable es José Vélez Serrano.

1.2 Alcance del sistema

El nombre del sistema será ImperDetect, teniendo como principal tarea la clasificación de tres tipos de imperfecciones en superficies metálicas: inclusiones, parches y arañazos. El dispositivo recibirá imágenes con defectos, por lo que la detección de defectos no será parte del sistema. Adicionalmente, se podrá implementar la localización de los defectos. Los beneficios del sistema son la rapidez y eficiencia en la automatización de este proceso, permitiendo derivar cada pieza por una línea diferente dependiendo del tipo de defecto.

1.3 Definiciones, acrónimos y abreviaturas

Es importante partir de las definiciones siguientes:

- **Inclusions:** Tipo de defecto detectado en el sistema.
- **Patches:** Tipo de defecto detectado en el sistema.
- **Scratches:** Tipo de defecto detectado en el sistema.
- **Red neuronal:** término que hace referencia a la familia de modelos computacionales que permiten el cálculo dentro de un conjunto de unidades llamadas neuronas. La información de entrada atraviesa la red neuronal produciendo unos valores de salida. Este tipo de modelos permiten resolver problemas de regresión y clasificación, así como de detección y localización en imágenes.
- ***Dataset*:** conjunto de datos disponibles para el entrenamiento de una red neuronal. Habitualmente se suelen dividir entre conjunto de *train*, de *validation* y de *test*. El conjunto de entrenamiento (*train*) comprende los datos empleados en el ajuste de los parámetros de la red neuronal. Han de ser representativos del total de datos, por lo que normalmente se seleccionan aleatoriamente. El conjunto de validación (*validation*) se emplea después de cada iteración en el proceso de entrenamiento, para comprobar si se produce el sobreaprendizaje. Por último, los datos de prueba *(test)* sólo se emplean una vez finalizado el entrenamiento.
- ***Deep Learning*:** técnicas basadas en la utilización de una red neuronal artificial compuesta por un número de niveles jerárquicos. En el nivel inicial de la jerarquía la red aprende algo

simple y luego envía esta información al siguiente nivel. El siguiente nivel toma esta información sencilla, la combina, compone una información algo un poco más compleja, y se lo pasa al tercer nivel, y así sucesivamente.

Se definen a continuación los acrónimos y siglas empleados en el documento.

BB	Bounding Box
CNN	Convolutional Neural Network
GT	Ground Truth
IA	Inteligencia Artificial
OpenCV	Open Source Computer Vision
YOLO	You Only Look Once
VA	Visión Artificial

1.4 Referencias

Standard IEEE 830 - 1998	IEEE
https://ieeexplore.ieee.org/abstract/document/9062515	Inspection of Imprint Defects in Stamped Metal Surfaces Using Deep Learning and Tracking (2020)
https://www.mdpi.com/1424-8220/20/19/5593	A Study of Defect Detection Techniques for Metallographic Images (2020)

1.5 Resumen

Este documento consta de tres secciones. En la primera sección se realiza una introducción al mismo y se proporciona una visión general de la especificación de recursos del sistema.

En la segunda sección del documento se realiza una descripción general del sistema, con el fin de conocer las principales funciones que éste debe realizar, los datos asociados y los factores, restricciones, supuestos y dependencias que afectan al desarrollo, sin entrar en excesivos detalles.

Por último, la tercera sección del documento es aquella en la que se definen detalladamente los requisitos que debe satisfacer el sistema.

2 Descripción general

En esta sección se describen los factores que pueden afectar al producto y a sus requisitos. No es una descripción de requisitos, sino una puesta en contexto.

2.1 Perspectiva del producto

El sistema ImperDetect está enfocado a cubrir los requerimientos de la empresa METALIKA, S.L. en la ciudad de Madrid, España. ImperDetect estará integrado dentro de una plataforma ya implementada en C de detección de defectos. Tiene por objetivo clasificar la imperfección entre tres categorías diferenciadas.

Se abordará el desafío mediante técnicas de *Deep Learning*, concretamente utilizando detectores de objetos como *YOLOv5*.

Es importante destacar dos aspectos. En primer lugar, ImperDetect es independiente de otros sistemas, sin embargo, cualquier fallo en la plataforma previa de detección tendrá su efecto en la respuesta. En segundo lugar, se debe poner atención en reducir el tiempo de ejecución por muestra, puesto que la empresa desea cumplir unos plazos con alto rendimiento.

2.2 Funciones del producto

ImperDetect recibirá una imagen del sistema previo de detección. Se adjuntan algunas imágenes de ejemplo de los defectos:

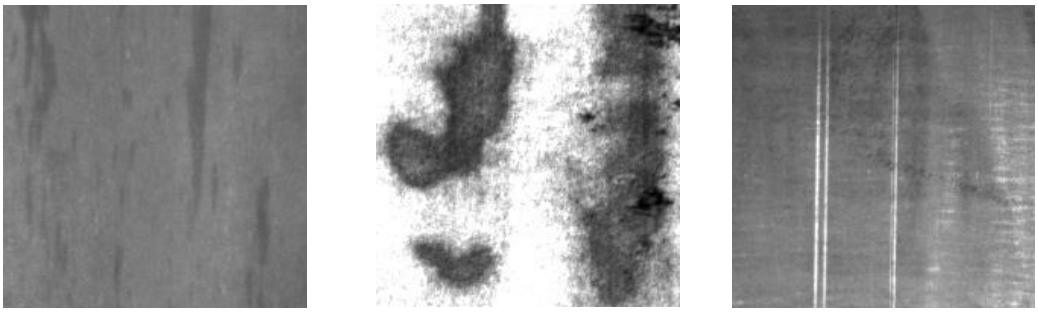
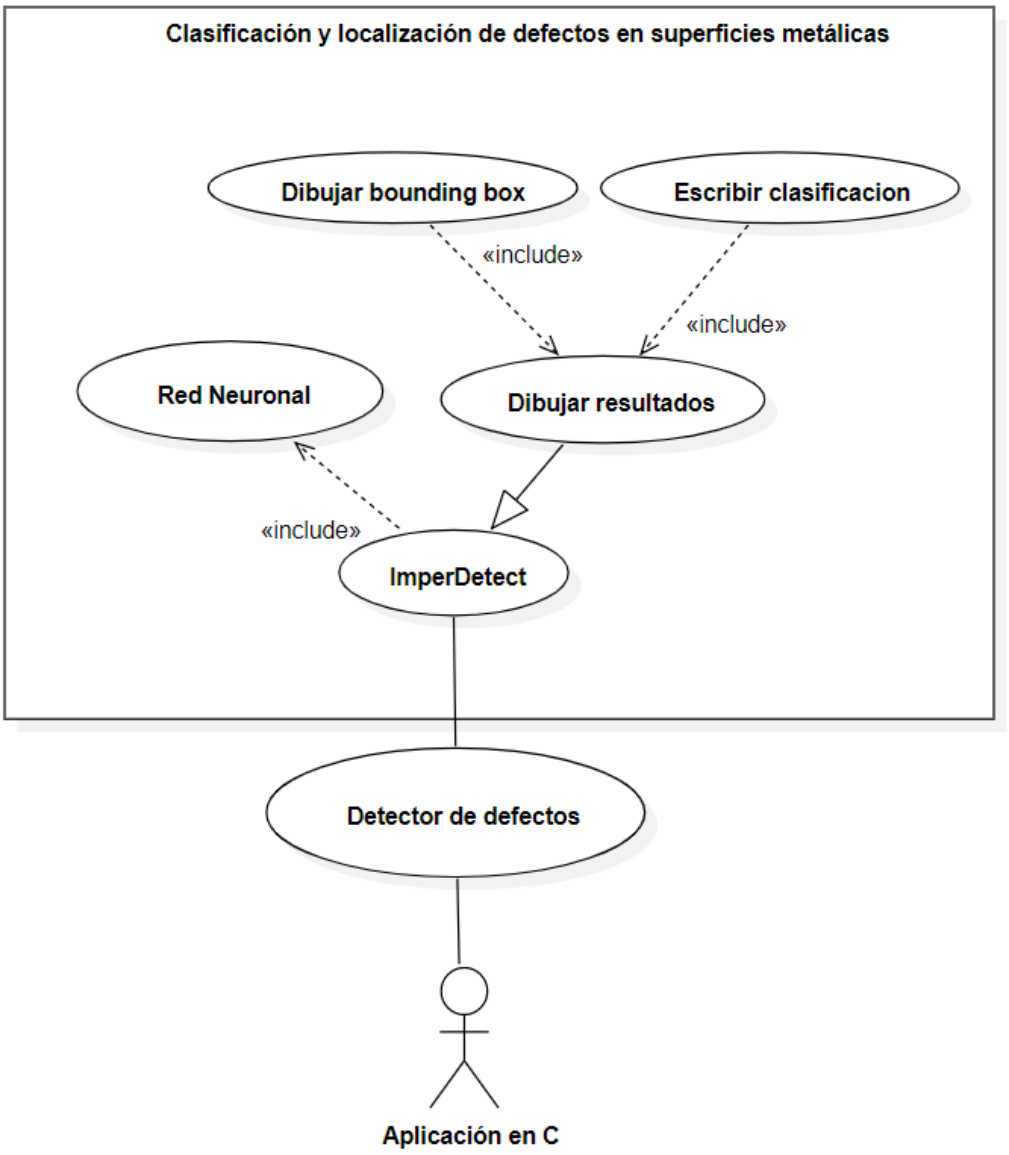


Imagen 1. Ejemplo de los defectos de izquierda a derecha: Inclusiones, patches y scratches.

Con estas imágenes nuestro sistema de *Deep learning* realizará la clasificación y la localización de los defectos. La salida del sistema será una clasificación de la imagen, y la localización de defectos en coordenadas. También se devolverá una imagen de menor calidad con la clasificación y *bounding box* para visualización en el HMI si el cliente lo desea.

A continuación, se muestra un diagrama de casos de uso del sistema, donde se puede ver manera simplificada el funcionamiento que tendrá el sistema:



Esquema 1 – Diagrama de casos de uso de la aplicación ImperDetect

2.3 Características del usuario

En un principio la aplicación no va a tener usuario que la controlen, simplemente el programador del sistema previo realizara una llamada a nuestra librería para obtener la clasificación y la posición.

Tipo de usuario	Programador cliente
Formación	Ingeniero
Actividades	Programación, integración e implementación de código.

2.4 Restricciones generales

- Lenguaje de programación.
  - Se utilizará Python, utilizando librerías muy potentes en visión artificial como son *OpenCV* y *Torch* para los modelos *Deep learning*.
  - Se utilizará C, creando una librería para realizar la llamada a nuestro programa en Python.
- Limitaciones del hardware. Para ejecutar la aplicación en tiempo real será necesario el servidor ya instalado en la fábrica. Se le añadirá una tarjeta gráfica RTX 2060 para el procesamiento en paralelo del algoritmo.
- Limitaciones de tiempo. Una de las limitaciones es el funcionamiento en tiempo real de la aplicación para poder realizar la clasificación adaptada a la velocidad de la línea.
- Limitaciones de acierto. La clasificación manual por operarios está cerca del 90% de acierto, por lo que debemos superar este porcentaje.

2.5 Suposiciones y dependencias

- Para el desarrollo del producto ImperDetect se cuenta con un pequeño *dataset* proporcionado por el cliente. Se dispone de 300 imágenes en *grayscale* de 200x200 pixeles por cada categoría a clasificar. También se cuenta con las etiquetas de estos defectos con archivos .xml. En total son 900 imágenes con sus correspondientes etiquetas.
- La aplicación se va a desarrollar en Python 3.9, usando bibliotecas como: *OpenCV* y *Torch*.

3 Requisitos específicos

En esta sección se detallan los requisitos con una profundidad suficiente para que los diseñadores puedan implementar el sistema y que el equipo pueda planificar las pruebas que demuestren el correcto funcionamiento del sistema.

3.1 Funciones

1. **Clasificación y localización**

   Función encargada de recibir una imagen donde contiene uno de los defectos propuestos por el cliente. Utilizando la red neuronal profunda YOLOv5 se realizará la inferencia sobre la imagen para obtener un resultado. De esta función devolveremos un resultado de clasificación y un *bounding box* de la localización.

   Para el desarrollo, entrenamiento y testeo se necesitan el número máximo de muestras posibles que el cliente nos pueda proporcionar. Como primera aproximación se ha entrenado un sistema con las 900 muestras proporcionadas en esta primera fase.

2. **Dibujar resultados:** Función encargada de una vez detectados los defectos dibujarlos sobre la imagen si el cliente lo desea. Esto puede ser útil para la depuración o el visionado por parte de los operarios.

3.2 Rendimiento

- El cliente no aceptará un sistema con un error superior al 10%, para superar la clasificación manual de los operarios.
- La velocidad de respuesta que espera el cliente del sistema tendrá que ser lo más cercana al tiempo real, donde cada imagen debe ser procesada en menos de un segundo. Puede existir una cola de procesado, pero teniendo en cuenta que no se alargue a lo largo de un día de producción.

3.3 Restricciones de diseño

- Se va a necesitar estudiar la forma en la que se va a establecer la llamada desde C a nuestra aplicación en Python.
- Se va a utilizar Programación Orientada a Objetos (POO) ya que permitirá que los desarrolladores puedan trabajar en distintas partes del programa.
- ImperDetect es independiente de otros sistemas, sin embargo, cualquier fallo en la plataforma previa de detección tendrá su efecto en la respuesta de nuestro sistema.

3.4 Atributos del sistema

- Portabilidad del desarrollo a otros entornos hardware o software. El cliente podrá acceder al servidor desde cualquier sitio.
- Mantenibilidad del sistema con el paso del tiempo. El código desarrollado cumplirá las normas de estilo de la empresa (nomenclatura de variables y funciones, tamaño de las funciones, comentarios…).

4 Presupuesto

Presupuesto del proyecto: Clasificación y localización de defectos en superficies de metal.

Elaborado por: Natalia Lourdes Pérez García de la Puente, Vicente Gilabert Maño y Luis Rosario Tremoulet.

Fecha de inicio del proyecto: 23/03/2022

Duración del proyecto: 2 meses desde la fecha de inicio.

Tarea / Actividad	Elemento	Unidades	Coste por unidad	Coste Total
Estudio previo del proyecto				
	Elaboración del documento de requisitos del sistema (DRS)	1	150,00 €	150,00 €
	Diseño conceptual del sistema	1	200,00 €	200,00 €
			Subtotal 1	**350,00 €**

Personal				
	Implementación del sistema.	3	2.000,00 €	6.000,00 €
			Subtotal 2	**6.000,00 €**

Hardware servidor				
	Tarjeta gráfica NVIDIA.	1	450,00 €	450,00 €
	Otros componentes	1	200,00 €	300,00 €
			Subtotal 3	**650,00 €**
			TOTAL (sin IVA)	**7.000,00 €**
			TOTAL (con IVA 21%)	**8.470,00 €**

Tarea / Actividad	Elemento	Unidades	Coste por unidad	Coste Total
Mantenimiento				
	Mantenimiento de los software y hardware. (anual)	1	3.500,00 €	3.500,00 €
			Subtotal 1	3.500,00 €
			TOTAL (sin IVA)	**3.500,00 €**
			TOTAL (con IVA 21%)	**4.235,00 €**