UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Clasificación y detección de errores en piezas para inspección industrial

TRABAJO FIN DE MÁSTER

Máster en Ingeniería Informática

Autor: Omar del Tejo Catalá

Tutor: Juan Carlos Pérez Cortés

Curso 2019-2020

Resumen

Se presenta una comparativa entre distintos métodos para la extracción de características de un conjunto de piezas industriales. El objetivo de estas características es entrenar un algoritmo de clasificación con rechazo capaz de determinar la clase a la que pertenece una nueva pieza, o si esta presenta una variación demasiado grande como para ser aceptada como válida. El propósito de la comparativa es escoger el mejor conjunto de características que mejore el sistema de clasificación con rechazo de Zerogravity3D. Este es un producto desarrollado dentro del Instituto Tecnológico de Informática para el campo de la inspección industrial. Obtiene un conjunto de 16 vistas desde distintos ángulos del objeto bajo inspección y realiza una reconstrucción del modelo. Actualmente utiliza únicamente las características de área, volumen y desviación típica de los puntos del modelo 3D.

Se comparan extractores de características tanto sobre los modelos 3D de las piezas como sobre sus vistas. Sobre los modelos 3D se extraen características geométricas ya existentes y se propone incluir un histograma de distancias de los puntos al centroide; y sobre las vistas del objeto se utilizan algunas técnicas clásicas de extracción de características de imagen (Zernike y RLBP) y *Deep Learning* (MVCNN).

De entre todos los extractores de características, Zernike y MVCNN son los que mejores resultados obtienen, con una tasa de acierto del 89.8 % y 93.6 % y una tasa de error de 4 % y 6.9 %, respectivamente. MVCNN, debido a que es un algoritmo de *Deep Learning*, es más costoso computacionalmente que Zernike y por eso tarda el triple que este (400 ms para MVCNN y 120 ms para Zernike en un procesador Intel(R) Xeon(R) CPU E5-2630 v3). Las características del modelo 3D propuestas obtienen una mejora del 8.7 % de precisión y reducen la tasa de error un 0.86 % frente a las ya existes. Sin embargo, no consiguen un acierto comparable con las dos técnicas mencionadas anteriormente (un 70.84 %), aunque obtienen una tasa de error baja del 4.05 %. Además, esta técnica requiere de una reconstrucción 3D, que es una etapa muy costosa del proceso, lo que hace que sea el extractor de características más lento (2 segundos). RLBP consigue menor tasa de acierto (un 67.53 %) y mayor tasa de error (un 14.7 %), sin embargo es el extractor de características más rápido, requiriendo únicamente 10 ms. La mejora en los resultados utilizando las vistas directamente permiten reducir el coste temporal de clasificación ya que no es necesaria la reconstrucción 3D.

Posteriormente se realiza una comparativa entre las dos mejores opciones (Zernike y MVCNN). Debido a factores como el coste de cómputo y riesgos a la hora de entrenar MVCNN, se escoge Zernike como la mejor alternativa. Con esto se propone una solución final y, finalmente, se describe el proceso de implantación

dentro del sistema Zerogravity3D.

Palabras clave: Clasificación, clasificación con rechazo, piezas industriales, inspección industrial, extracción de características, Redes Neuronales, MVCNN, Zernike, RLBP, modelos 3D

Abstract

This work presents a comparison between feature extractors over a set of industrial parts. The goal of this features is to train a model able to clasify a part into its corresponding class, or discard the part if it contains a significant deviation. The purpose of this comparison is to select the most significant features to improve the clasification with rejection system within Zerogravity3D. This is a product developed by the Technological Institute of Computer Science. Obtains a set of 16 views from the part under inspection and reconstructs it. Currently, it extracts the geometrical features area, volume and standard deviation from the points of the 3D model.

Both 2D view features and 3D model features are considered. The histogram of distances to the centroid of the model is proposed to be included along with the existing 3D features extracted. For the views of the part, some classical techniques (such as Zernike and RLBP) and *Deep Learning* (MVCNN) are considered.

Among all feature extractors, Zernike and MVCNN achieved the best results, with a 89.6% and 93.6% precision and 4% and 6.9% error rate, respectively. MVCNN, due to the fact that is a *Deep Learning* technique, is more computationally expensive than Zernike and therefore takes three times more time to compute (400 ms for MVCNN and 120 ms for Zernike in an Intel(R) Xeon(R) CPU E5-2630 v3 processor). The 3D features proposed achieve a 8.7% increase in precision and a 0.86% reduction on the error rate. However, it doesn't achieve a precisión (70.84%) comparable to the previous features, although it gets a low error rate (4.05%). Moreover, it is the slowest of all feature extractors (2 seconds), as it requires to perform the 3D reconstruction. RLBP obtains the lowest precision (67.53%) and a higher error rate (14.7%) but it is the quickest, needing just 10 ms to compute the features. The fact that the best results are the ones using the views of the parts allows the removal of the 3D reconstruction phase.

Furthermore, the two best options, Zernike and MVCNN, are compared. Due to the computational cost and training risks of MVCNN, Zernike is chosen as the best alternative. This is proposed as a final solution to the project and, finally, the instalation process inside Zerogravity3D is detailed.

Key words: Classification, industrial parts, industrial inspection, feature extraction, neural networks, MVCNN, Zernike, RLBP, 3D models

Índice general

Índice de figuras

Índice de tablas

CAPÍTULO 1

Introducción

El campo de la clasificación automática es de gran interés para la sociedad. Tiene multitud de aplicaciones posibles: desde diagnosticar posibles enfermedades del hígado examinando la pigmentación de la piel, hasta desbloquear automáticamente los dispositivos móviles mediante la cámara frontal. La investigación en este campo ha ido avanzando en paralelo a las mejoras en las capacidades de cómputo, alcanzando su época de esplendor en las últimas dos décadas.

La quintaesencia de la clasificación es la selección de una serie de elementos representativos, también llamados características, que son propios del elemento que se busca clasificar, y distintos de cualquier otro elemento existente. Estas características pueden variar entre objetos de una misma clase: existen bolígrafos de distinto tamaño y color, pero ambos son bolígrafos. Pero, ¿cuánto se debe deformar un bolígrafo para que deje de poder ser considerado como tal? Por supuesto, la extracción de las mejores características que describan a un objeto están sujetas a cuestiones de teoría del lenguaje, que no se suelen abordar formalmente para cada problema. En cambio, se suelen utilizar técnicas genéricas de extracción de características dependiendo del contexto de este.

La inspección automática es una de las tareas más importantes dentro de las líneas de producción de las fábricas. Estas deben incorporar algún control de calidad, físico o químico, de las piezas desarrolladas con el fin de descartar aquellas que no sean válidas para la comercialización. Por esto, es importante que los algoritmos de clasificación, no solo clasifiquen correctamente las piezas dentro de la clase que corresponda, sino que tengan la capacidad de descartar aquellas piezas que no cumplan los requisitos mínimos. Por ejemplo, se espera que un tornillo tenga unas dimensiones características para poder ser comercializado. Si las di-

mensiones son muy distintas a lo esperado, este debe ser descartado.

En la mayoría de los casos, la inspección física automática se realiza mediante cámaras, que buscan grandes variaciones con respecto a una pieza modelo. Existen múltiples aproximaciones a este problema. Por ejemplo, si el objetivo es determinar si un objeto es defectuoso dada una clase (es decir, su clase ya es conocida de antemano), se puede extraer información de la textura del objeto y su forma para determinar si es una pieza defectuosa [4][3][20]. Por otra parte, este trabajo realiza una aproximación distinta: se busca obtener un algoritmo que clasifique las piezas dentro de sus respectivas clases y, a la vez, sea capaz de rechazar aquellas piezas que sean muy distintas a las piezas originales. Esta clase de problemas se engloban dentro de la clasificación con rechazo.

Este trabajo está siendo desarrollado dentro del marco de trabajo del Instituto Tecnológico de Informática. La base de esta investigación está formada por una tecnología de inspección industrial desarrollada por este, llamada Zerogravity3D, que será explicada con profundidad en la sección "*2.2. Zerogravity3D*". Esta obtiene 16 vistas de un objeto que es lanzado al aire, eliminando las caras ocultas y permitiendo una inspección completa del objeto en un solo lanzamiento.

1.1 Motivación

La inspección industrial manual es costosa debido a la necesidad de contratar personal especializado para la tarea, y tedioso para el propio trabajador. Los controles de calidad son un requisito que deben cumplir las empresas que buscan comercializar un producto, por lo que los avances en este campo suponen un gran impacto para la industria. Este proceso no debe convertirse en un cuello de botella y, por esto, en la mayoría de los casos suele estar automatizado; el caso más común es el reconocimiento de imperfecciones en las piezas por imagen que se realiza utilizando cintas transportadoras.[1][2] Sin embargo, esto implica que existe una zona oculta de las piezas que no se comprueba en la primera iteración e implica realizar pruebas complementarias.

Hasta la fecha, la mayoría de las técnicas de inspección industrial realizadas con Zerogravity3D utilizaban los modelos 3D generados a partir de las imágenes. Sin

[1] https://www.rnaautomation.com/products/vision-inspection-systems/
[2] https://lmi3d.com/factorysmart/part-manufacturing-inspection

embargo, en este proyecto se busca eliminar la costosa etapa de reconstrucción 3D y permitir una inferencia sobre las vistas obtenidas del objeto. Además, la etapa de reconstrucción debe hacerse con todas las imágenes y de forma centralizada, por lo que el hecho de eliminar la esta etapa tiene la ventaja añadida de que cada imagen puede ser procesada en paralelo, aumentando la cantidad de piezas que pueden ser procesadas por segundo.

Actualmente Zerogravity3D es un prototipo, por lo que se encuentra en una fase de búsqueda de mejoras continua con el fin de aumentar el valor del producto. Es necesario poner a prueba las características que se seleccionan actualmente para Zerogravity3D con bases de datos más complejas de las que se habían evaluado hasta el momento para simular posibles escenarios complejos en las fábricas. Además, estas características son considerablemente genéricas (área, volumen y desviación típica) por lo que encontrar otro conjunto de características, o mejorar el existente, puede incrementar la calidad de las estimaciones.

Por otra parte, ha habido grandes avances en el tratamiento de las imágenes y el reconocimiento de formas a lo largo de los últimos años. Las competiciones de clasificación y detección de imágenes están siendo lideradas por técnicas que se engloban dentro del *Deep Learning*. Los nuevos avances en este campo se han visto apoyados por mejoras en el hardware de los sistemas de cómputo, lo que ha permitido que los elevados requisitos computacionales de algunas de estas técnicas no sean un impedimento para su uso. Estas mejoras implican que haya habido una mayor implantación de estas técnicas en la industria.

1.1.1. Motivación personal

Las técnicas de *Deep Learning* aplicadas a cualquier problema me apasionan especialmente. Han demostrado a lo largo de los últimos años un potencial sorprendente; tanto que la mayoría de los productos cotidianos han empezado a incorporarlas. Los avances en clasificación dentro del campo de medicina para el diagnóstico de enfermedades me resultan particularmente interesantes por el gran potencial que tienen para mejorar la calidad de vida de la sociedad y, por ejemplo, democratizar la salud.[3]

[3]Recientemente se han realizado estudios acerca de la diagnosis rápida de una serie de enfermedades a través del teléfono móvil [17]. El propósito no era sustituir el criterio de un profesional, si no más bien advertir al usuario si debería realizarse alguna inspección.

Es abrumador la cantidad de esfuerzo que se invierte en este campo, y lo interesantes que resultan las propuestas realizadas por otros investigadores. Otra ventaja es que es bastante accesible a nuevas personas interesadas, gracias a que existen múltiples librerías que simplifican los conceptos matemáticos de este campo. Sin embargo, el mayor problema siguen siendo los requisitos computacionales del *Deep Learning*.

1.2 Objetivos

El objetivo principal de este proyecto es añadir valor al producto Zerogravity3D, en concreto mejorando su capacidad de clasificación de las piezas dentro de una industria. Se realiza esto mediante la propuesta de una alternativa a su sistema de clasificación, en particular, de las características que extrae de los objetos que inspecciona. Este objetivo principal se puede desglosar, a su vez, en varios objetivos secundarios:

- Comparar distintos métodos para la extracción de características discriminativas del objeto inspeccionado, tanto sobre las distintas vistas 2D del modelo como sobre el modelo 3D. Para realizar esta comparativa se experimenta con el modelo de clasificación con rechazo existente de Zerogravity3D, que permite catalogar el objeto como una pieza de una determinada clase, una pieza desconocida para el sistema o como una pieza defectuosa (basándose en las variaciones asequibles que se observan en los objetos en buen estado).
- Evaluar la mejora frente al sistema de clasificación existente.
- Proponer la mejor alternativa como un nuevo sistema de clasificación.
- Elaborar un estudio de la implantación de esta propuesta dentro del producto: gestión de riesgos, análisis de presupuesto, diseño del plan de trabajo y la arquitectura de la solución.

1.3 Impacto esperado

Se espera obtener un algoritmo que mejore el sistema existente en Zerogravity3D, con el fin de incrementar su valor y que este sea más genérico y funcione correctamente en un caso de uso de una fábrica. Se busca mejorar la precisión de este algoritmo y minimizar el tiempo de clasificación con el fin de procesar más piezas por minuto. Con todo esto se pretende que Zerogravity3D aumente su valor

hasta sobrepasar a sus competidores, superándolos en precisión con un menor coste de procesamiento. Todo esto vinculado a las mejoras que aporta el diseño de Zerogravity3D, que permite una inspección holística de los objetos en un solo lanzamiento. Esto se explica en mayor detalle en la sección 2.2.

La tabla 1.1 recoge el análisis DAFO del proyecto y del producto esperado de este.

Debilidades	**Fortalezas**
▪ Uso de algunas técnicas que no son novedosas. ▪ Gran variabilidad en el tipo de piezas que se esperan poder ser clasificadas. ▪ Zerogravity3D es un prototipo sin modelo en producción que de información acerca dé posibles riesgos inesperados.	▪ El trabajo ha sido realizado dentro de un equipo de con grandes conocimientos en el campo. ▪ Zerogravity3D cuenta con varias patentes.
Amenazas	**Oportunidades**
▪ Que el sistema propuesto no pueda competir en precisión y tiempos con otros sistemas del mercado.	▪ Dar valor a los productos y a la investigación realizada dentro del ITI.

Tabla 1.1: Análisis DAFO.

1.4 Metodología

Debido a la naturaleza del proyecto, se ha podido dividir este en varias secciones. Cada extractor de características sobre las imágenes 2D, debido a su similitud, se estima que ocuparán el mismo tiempo de proyecto. MVCNN, sin embargo, al ser una red neuronal necesitará más tiempo debido a los enormes tiempos de cómputo que requiere su entrenamiento. En la figura 1.1 se muestra el diagrama de Gantt estimado para el proyecto. En rojo aparecen partes del proyecto que no han sido incluidas en el report final. En azul se muestran las que sí han sido incluidas. En verde se muestra la experimentación previa al proyecto, que motivó el inicio de este. Se realizaron algunas variaciones en el mes de julio para volver a estimar la planificación tras las pruebas con DCT, que resultaron ser inabarcables.

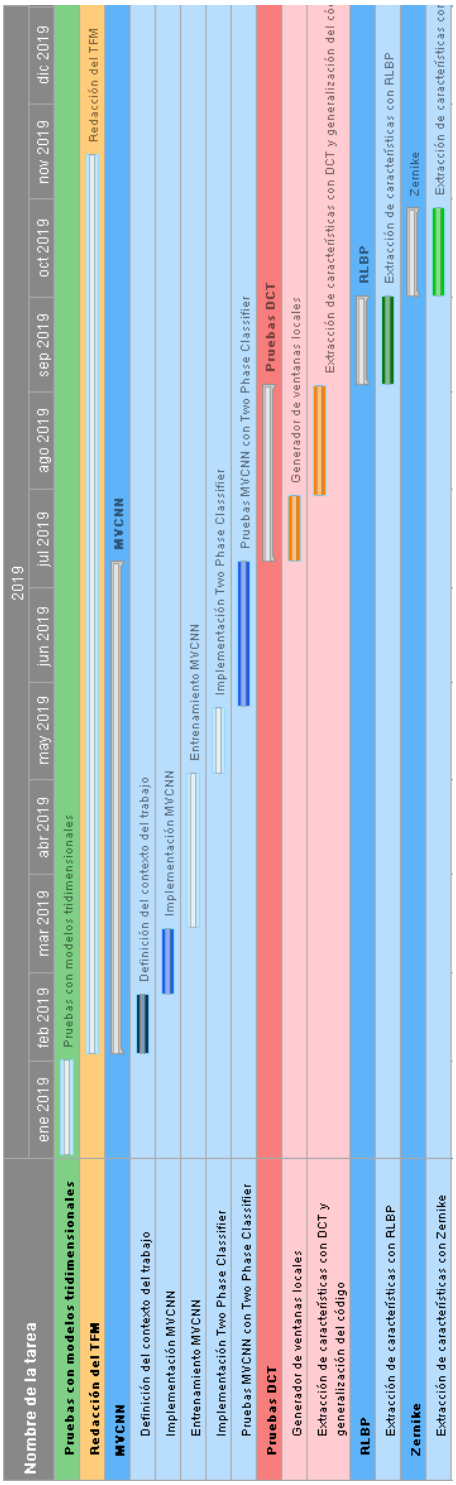


Figura 1.1: Diagrama de Gantt de la planificación del proyecto.

1.5 Estructura

El trabajo puede dividirse en dos bloques: un primero donde se describe la experimentación realizada y un segundo donde se propone una solución basada en los resultados obtenidos y se realiza un estudio de la implantación.

Los apartados en los que se divide el trabajo son los siguientes:

- *"1. Introducción"*.
- *"2. Contexto tecnológico"*.
- *"3. Análisis del problema"*.
- *"4. Experimentación"*.
- *"5. Comparativa"*.
- *"6. Análisis de riesgos"*.
- *"7. Solución propuesta"*.
- *"8. Conclusiones"*.
- *"9. Trabajo futuro"*.
- *"10. Relación con los estudios cursados"*.
- *"11. Agradecimientos"*.
- *"12. Glosario"*.

En el apartado *"2. Contexto tecnológico"* se presentan y se explican los conceptos y la terminología que serán utilizados a lo largo del trabajo, lo cual incluye un estudio de múltiples técnicas de extracción de características y de reconstrucción de modelos y una descripción del sistema Zerogravity3D. Enb la sección *"3. Análisis del problema"* se describe la estructura del proceso que sigue la experimentación. En el apartado *"4. Experimentación"* se describe en detalle la experimentación realizada y los resultados obtenidos. En el apartado *"5. Comparativa"* se analizan los resultados obtenidos para todos los experimentos y se pondera la mejor alternativa. En la sección *"6. Análisis de riesgos"* se analizan los riesgos de esta mejor alternativa y se realiza una propuesta fundamentada de la solución propuesta. En el apartado *"7. Solución propuesta"* se describen los requisitos, presupuesto y plan de trabajo para la implantación de esta propuesta. En la sección *"8. Conclusiones"* se realiza un resumen de la experimentación realizada y de la solución propuesta. En el apartado *"9. Trabajo futuro"* se describen los experimentos que no han

podido incluirse dentro de este trabajo pero que son interesantes de evaluar. En la sección *"10. Relación con los estudios cursados"* se realiza una introspección en busca del origen de los conocimientos aplicados a lo largo del trabajo. En el apartado *"11. Agradecimientos"* se recogen las personas sin las cuales este trabajo no habría podido ser realizado. Finalmente, en la sección *"12. Glosario"* se muestran los conceptos que no han sido explicados explícitamente a lo largo del trabajo y pueden ser desconocidos para el lector.

1.6 Convenciones

Como se ha mencionado en el apartado anterior, existe un glosario que contiene una descripción de los conceptos que se mencionan en el trabajo y no se explican de forma explícita. Todas estas palabras se marcan en cursiva para indicar que, si no se conociera el significado de estas, aparece en el apartado de *"12. Glosario"*.

CAPÍTULO 2

Contexto tecnológico

2.1 Reconstrucción 3D de objetos

La reconstrucción 3D de objetos mediante imágenes 2D es una técnica ampliamente utilizada en multitud de campos como, por ejemplo, para el diagnóstico de cáncer de mama [23] [9]. La idea es recuperar la información acerca de la profundidad de los escenarios que se pierde al reducirlos a 2D. Esto implica que las reconstrucciones no pueden hacerse únicamente con una imagen, y que se necesitan, teóricamente, al menos dos imágenes para poder obtener la profundidad de los puntos de las imágenes. Estas imágenes pueden ser obtenidas por múltiples cámaras en el mismo instante de tiempo (el caso de Zerogravity3D) o mediante una misma cámara en distintos instantes de tiempo.

Para este proceso es necesaria una correcta calibración de los parámetros intrínsecos y extrínsecos de las cámaras. Los parámetros intrínsecos son aquellos propios del conjunto de ópticas y sensor de cada cámara, como por ejemplo la distancia focal. Para la calibración de estos parámetros se utilizan patrones conocidos, como por ejemplo un tablero de ajedrez el cual, debido a su estructura regular, permite identificar y corregir posibles efectos de distorsión de las lentes como la distorsión de barril. Varios ejemplos de esta distorsión se pueden ver en las figuras 2.1 y 2.2. La librería OpenCV incluye algoritmos que permiten realizar esta calibración automáticamente a partir de varias imágenes desde múltiples ángulos de un tablero de ajedrez.

Los parámetros extrínsecos son aquellos que configuran la posición de la cámara dentro del mundo, como por ejemplo las coordenadas X, Y y Z de la cámara en la escena. Esta calibración nos permitirá pasar de los puntos locales de las cámaras

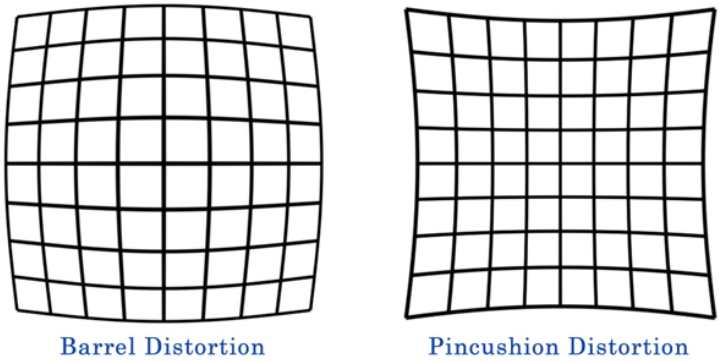


Figura 2.1: Ejemplos de distorsión en las lentes.

Figura 2.2: Ejemplos de distorsión en las lentes.

a los puntos globales de la escena.

El siguiente problema consiste en buscar puntos en las imágenes que concuerden entre sí. De esta forma, teniendo un mismo punto en dos imágenes distintas, podemos sacar la profundidad de este (ver imagen 2.3). Para esto se proyecta un rayo desde la cámara hasta cada píxel del contorno del objeto capturado. Este haz se entrecruza con haces de otras cámaras y permiten describir la forma del objeto. Un ejemplo de esto se puede visualizar en la figura 2.4.

Con esto se puede extraer la nube de puntos de cada vista y, posteriormente, combinarlas para generar una única nube de puntos que constituye el modelo 3D reconstruido. A lo largo del trabajo este nuevo modelo generado a partir de las vistas de las piezas será referido como reconstrucción o lanzamiento.

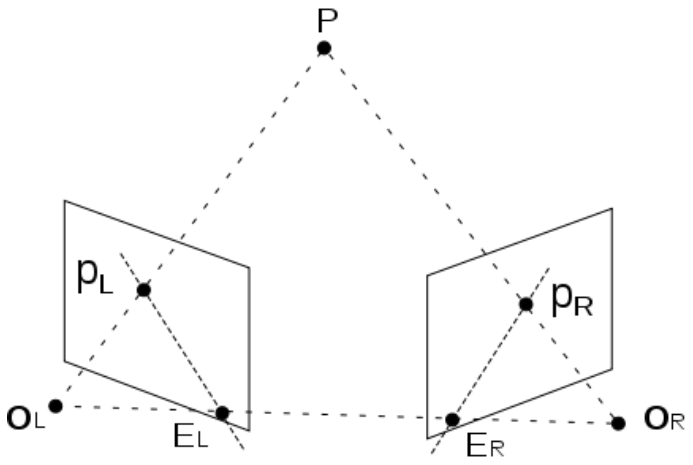


Figura 2.3: Obtención de la profundidad de un punto mediante múltiples cámaras [22].

2.2 Zerogravity3D

Zerogravity3D [1][2] es un sistema de inspección industrial desarrollado por el Instituto Tecnológico de Informática que permite obtener un conjunto de 16 vistas de una pieza bajo inspección. La figura 2.5 muestra el poliedro con 17 caras de Zerogravity3D.

Cada una de estas caras contiene una cámara que apunta al centro del poliedro, donde se espera que aparezca la pieza inspeccionado. 16 de estas cámaras cap-

[1] https://www.iti.es/soluciones/Zerogravity3D-3d/
[2] https://www.youtube.com/watch?v=5l7ajodJATE

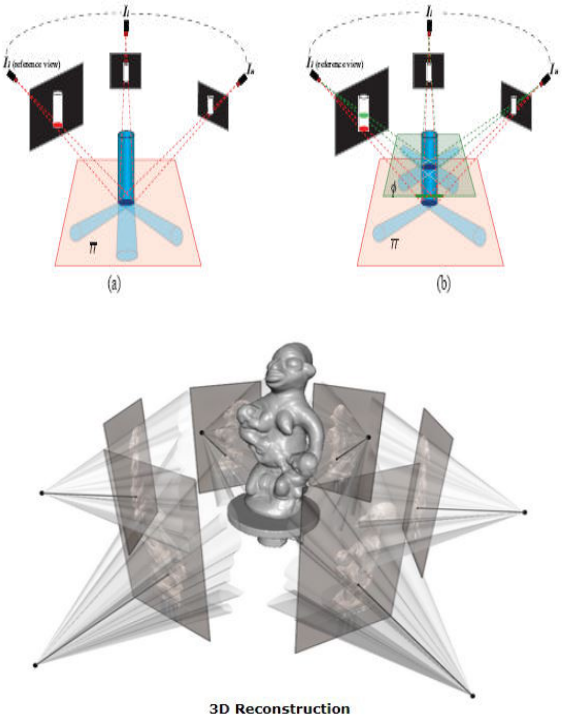


Figura 2.4: Obtención de la forma del objeto mediante rayos proyectados desde la cámara.

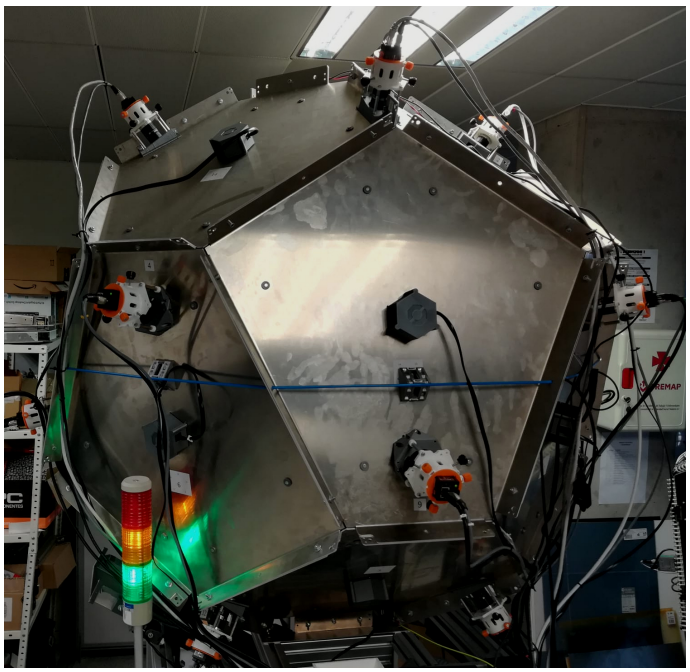


Figura 2.5: Zerogravity3D.

turan las múltiples vistas del objeto y la última, con una vista cenital, sirve para monitorizar el interior del poliedro. Se llama Zerogravity3D puesto que las imágenes del objeto son tomadas cuando la pieza está en caída libre.

De esto concepto de caída libre surgen dos variantes de este modelo: una en la cual las piezas son dejadas caer desde arriba y una segunda donde las piezas son lanzadas desde abajo hacia el centro del poliedro. La implementación actual es la segunda ya que permite reducir varios requisitos del sistema. Si la pieza se encuentra el movimiento, las cámaras deben de tener una velocidad de obturación mayor, lo cual implica una mayor iluminación y mayor gasto. Además, se requerirían unas cámaras más costosas que incrementarían el precio total del producto. También es más difícil controlar que la trayectoria de la pieza pase por el centro del poliedro.

En el primero modelo, las piezas circulan por una cinta hasta una apertura en la parte superior de Zerogravity3D donde son dejadas caer hacia el interior. Cabe destacar que en este modelo Zerogravity3D únicamente cuenta con 16 caras, ya que la cámara cenital superior de monitorización necesita ser eliminada para dejar una apertura. Una vez se estima que la pieza está en la zona central, se captura. Esta cae dentro de otra cinta transportadora que se encarga de separar la pieza según el resultado del clasificador.

En el segundo, las piezas circulan por una cinta transportadora hasta un cesto enganchado en brazo mecánico, que empuja la pieza hacia el interior de Zerogravity3D. Esta sale proyectada hacia el centro de la esfera, donde es capturada, y vuelve a caer dentro del cesto. El cesto se voltea y deja caer la pieza en una cinta. De la misma forma que en el modelo anterior, las piezas circulan por la cinta hasta un separador que las redirige según el resultado del clasificador.

La calibración extrínseca de las cámaras se realiza automáticamente mediante el uso de unas marcas fiduciales. Estas marcas se sitúan en un lugar fijo, al lado contrario de la cámara. Una vez calibradas, cada cámara captura las marcas fiduciales y es capaz de detectar si esta se ha movido comparando la posición de las fiduciales en capturas posteriores. Además es capaz de medir los ajustes que debe aplicar para corregir esta desviación sin necesidad de recalibrar.

Cuando el objeto es capturado, cada cámara envía la información de la captura a un nodo centralizado. Allí se genera la reconstrucción a 3D para ser clasificado. En el futuro se planea complementar el nodo centralizado con varias placas, una asociada a cada cámara, para que sea capaz de preprocesar las imágenes antes del envío y reducir la carga en el nodo central. Todos los tiempos que se recogen en este trabajo están medidos en este servidor central, que está formado por 32 procesadores Intel(R) Xeon(R) CPU E5-2630 v3.

Zerogravity3D está diseñado para realizar múltiples tareas de evaluación de la calidad que son interesantes dentro del contexto industrial:

- Permite la clasificación de un lote de piezas, con el fin de detectar posibles piezas anómalas dentro del lote.
- Permite inspeccionar cada pieza individualmente para determinar si la pieza cumple con unos requisitos de calidad establecidos.

Además, Zerogravity3D cuenta con una web desde donde un operario puede ejecutar, monitorizar y evaluar cualquiera de los trabajos para los que ha sido diseñado el sistema. Esta es el punto de control de todo el proceso por lo que debe de contar con toda la funcionalidad de Zerogravity3D:

- Calibrar las cámaras antes del proceso.
- Modificar el número de intentos en los que cada pieza intenta ser clasificada.
- Asignar etiquetas al lote.
- Comprobar el número de piezas procesadas y cuantas de estas has sido procesadas correctamente.
- Comprobar qué piezas han sido clasificadas como defectuosas.
- Visualizar las capturas de cada una de las piezas del lote y su reconstrucción 3D.

La figura 2.6 muestra la visualización 3D desde la web de una pieza procesada.

2.3 Extractores de características de imagen

Se deben presentar ahora las técnicas clásicas, y otra basada en *Deep Learning*, de extracción de características en imágenes que van a ser utilizadas en el trabajo.

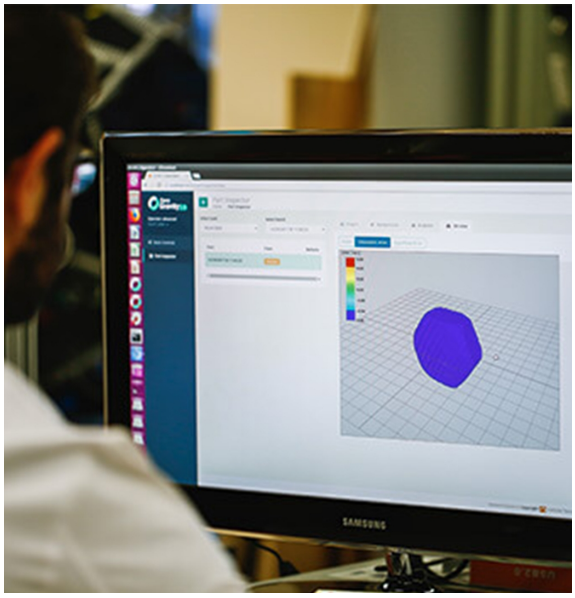


Figura 2.6: Web de monitorización del proceso de Zerogravity3D.

Para este problema es importante que estas características sean invariantes a la rotación y a la traslación, pero no al escalado, ya que puede haber piezas de distinto tamaño que se quieran diferenciar. Debido a que Zerogravity3D proyecta las piezas hacia su interior para ser capturada desde múltiples ángulos, la posición de la pieza no debe de influir en la extracción de las características.

La pieza puede rotar en cualquier ángulo dependiendo de factores que no se pueden controlar actualmente, como la posición de la pieza dentro del cesto o si la pieza conserva alguna velocidad angular antes de ser proyectada debido a la caída desde la cinta que la trasportaba. Estos factores también influyen en que la pieza no sea proyectada exactamente al centro de Zerogravity3D, por lo que la posición tampoco debe ser determinante. Además, para el caso de las características de imagen, se requiere recortar la imagen para reducir el consumo de memoria. Esto hace que cualquier información espacial se pierda, pero esta no es necesaria para la extracción de características de la imagen. En cambio, este recorte no puede ser realizado para la extracción de características 3D.

Se consideran los siguientes extractores de características: los momentos de Zernike y Rotated Local Binary Patterns (RLBP) dentro de las técnicas clásicas, y Multi-View Convolutional Neural Network (MVCNN) dentro de las técnicas de *Deep Learning*.

La obtención de características de imagen puede ser dividida entre características locales o globales [15]. Las características locales son extraídas de ventanas de menor tamaño de la imagen. En cambio, a las características globales que se obtienen de la imagen completa. El tamaño de la base de datos (presentada posteriormente en la Sección 4.1) no aconseja utilizar ventanas locales ya que, al haber un gran número de clases distintas, el número de ventanas locales extraídas incrementa de forma considerable. Por ello, se utilizan los extractores de características MVCNN, RLBP y Zernike sobre las imágenes completas. RLBP y Zernike extraen características para cada una de las vistas del objeto 3D y se considera que cada una es capaz de representar la clase a la que pertenece el objeto 3D. Sin embargo, MVCNN fusiona las características globales de las múltiples vistas del objeto para obtener una única característica global.

2.3.1. Técnicas clásicas

Rotated Local Binary Patterns (RLBP)
LBP [19] es una técnica de extracción de características que convierte una imagen en un histograma de patrones en la escala de grises. Esta técnica recorre la imagen obteniendo ventanas de un determinado tamaño. Extrae el valor del píxel central de la ventana y convierte el resto de valores a binario asignando el valor 1 si el valor de la celda es mayor que el valor del píxel central o 0 si es menor. Posteriormente, estas ventanas se convierten a vectores unidimensionales. Para ello, se selecciona una celda relativa al centro como valor inicial y se escogen los valores en el sentido de las agujas del reloj. Por ejemplo, seleccionando como valor inicial la esquina superior izquierda, la matriz [100 25 28; 167 115 159; 230 90 72] se convierte en el vector [00011100] (el valor central se excluye).

Se determina como patrón uniforme aquel que tiene como máximo dos transiciones de 0's a 1's y de 1's a 0's. Por ejemplo, [0010000] sería un patrón uniforme ya que solo tiene dos transiciones y [10101000] no sería uniforme ya que tiene 5 transiciones.

Una vez se obtienen todos los vectores de patrones de la imagen se genera un histograma con los 58 patrones uniformes posibles y otra barra adicional para todos los no uniformes.

Esto sería el bosquejo del algoritmo de LBP clásico. Sin embargo, este algoritmo no es invariante a la rotación. Por ello, se han propuesto variaciones al algoritmo

para conseguir esta invarianza [18] [1]. En el caso de [1] se obtiene la invarianza modificando el histograma final mediante la transformada de Fourier, haciéndolo invariante a rotaciones. En el caso de [18] se consigue en la fase de selección del orden del vector. Como se ha comentado, LBP se selecciona una posición relativa constante a la hora de construir el vector. Sin embargo, en [18] se propone obtener el orden del vector mediante la diferencia máxima entre el valor de las celdas y el centro de la ventana. Esta última es la aproximación que se utiliza en este trabajo.

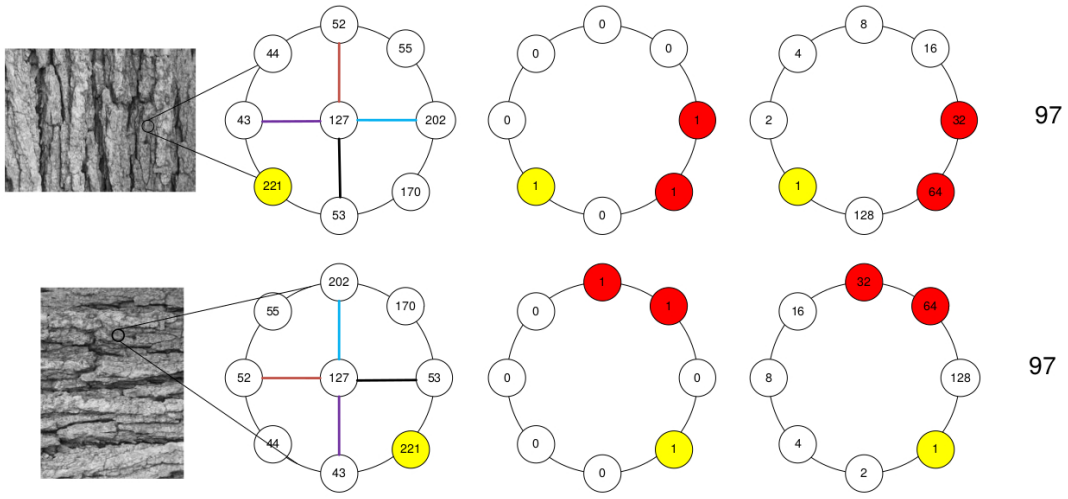


Figura 2.7: Ejemplo de la invarianza ante la rotación de RLBP [18]. En amarillo el píxel de mayor valor. En rojo los píxeles que tengan un valor superior al central.

Momentos de las imágenes

Es una característica de las imágenes que se obtiene de realizar una suma ponderada de las intensidades de sus píxeles. Las variaciones en la forma de ponderar de estas intensidades dan lugar a múltiples tipos de momentos de las imágenes. En este trabajo nos centraremos únicamente en dos: el momento base de las imágenes y los momentos de Zernike.

Momento base de una imagen

Dada una función $I(x, y)$, que devuelve la intensidad de las coordenadas (x, y) de una imagen, se describe el momento base de una imagen M_{ab} de la siguiente forma:

$$M_{ab} = \sum_{x} \sum_{y} x^a y^b I(x, y) \tag{2.1}$$

Estos momentos son especialmente relevantes para calcular el centroide de la imagen. Por ejemplo, en una imagen que se busca segmentar, es importante de-

tectar dónde se encuentra el centro del objeto ($\tilde{x}$,$\tilde{y}$), para segmentar alrededor de este. Esto se calcula mediante las fórmulas:

$$\tilde{x} = \frac{M_{10}}{M_{00}} \tag{2.2}$$

$$\tilde{y} = \frac{M_{01}}{M_{00}} \tag{2.3}$$

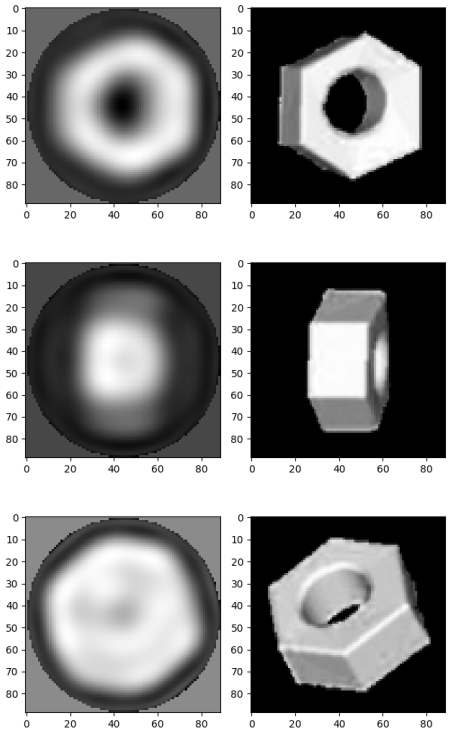
Momentos de Zernike
Esta es una técnica de extracción de características, descrita en profundidad en [13], que permite extraer características de imagen invariantes a la rotación. Extrae los momentos de la imagen, de forma similar a los momentos de Hu, derivados de los momentos base de la imagen explicados en el apartado anterior, que han sido tradicionalmente utilizados para clasificación de imágenes.

Estos momentos se calculan mediante los polinomios complejos ortogonales que describen el círculo unitario central de la imagen. Cuanto mayor sea el número de polinomios que se utilicen (esto es, cuantos más momentos se extraigan), mejor se puede aproximar la imagen en su zona central. Los píxeles que estén fuera de este círculo central no se utilizan para la extracción de características y por tanto se descartan.

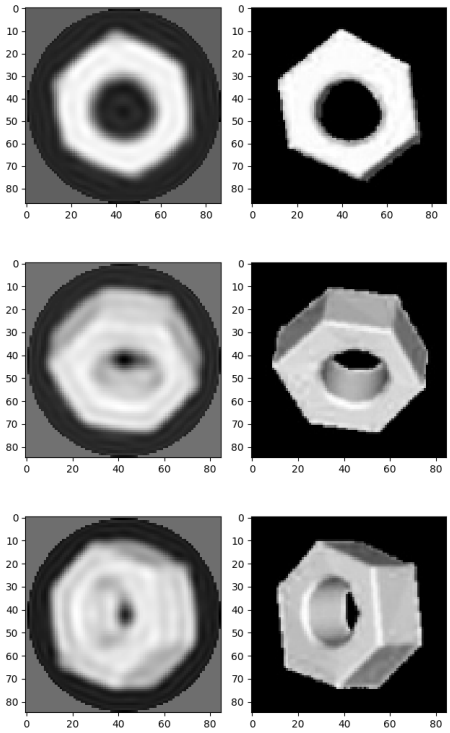
Lo que se utiliza como características de la imagen (los momentos) son los escalares complejos que están vinculados a estos polinomios. Tal y como se demuestra en [13], estas características extraídas son invariantes a la rotación cuando se extrae la magnitud de los escalares complejos, ya que una rotación en la imagen únicamente se corresponde a un cambio en la fase de estos. Un ejemplo de como afecta el número de momentos de Zernike extraídos en la reconstrucción de las piezas se puede ver en la figura 2.8.

2.3.2. Deep Learning

Convolutional Neural Network (CNN)
A lo largo de los últimos años, este tipo de redes neuronales ha ido ganando relevancia debido a sus resultados principalmente en el campo de visión por computador. En parte, este crecimiento se ha visto favorecido por el aumento de la capacidad de cómputo, en concreto, de los procesadores gráficos. Los procesadores



(e) 49 momentos



(f) 256 momentos

Figura 2.8: Reconstrucciones de piezas utilizando distinto número de momentos de Zernike sobre una arandela. Cada par representa, en el lado izquierdo, la reconstrucción del objeto a partir del tamaño de momentos de Zernike escogidos y, a la derecha, la imagen original.

gráficos están diseñados y preparados para realizar operaciones matriciales más rápidas que en las CPUs. Los aumentos tanto en el número de núcleos, donde se realizan las operaciones en paralelo, y el aumento en tamaño y velocidad de la VRAM han favorecido enormemente que esta clase de redes tengan una mayor implantación en la industria y se motive la investigación sobre ellas.

Las CNNs, como cualquier otra red, extraen características de la entrada a esta. Lo realizan a lo largo de múltiples capas; en concreto, en las CNNs se extraen las características de la capa anterior (o si fuera la primera capa de la red, de la entrada) mediante el operador de convolución. Este operador consiste en multiplicar un conjunto de variables (en redes neuronales se llaman pesos, aunque en esta clase de redes también se pueden llamar ventana) a lo largo de una señal, la cual, en el caso de este trabajo, es una imagen. La figura 2.9 muestra un ejemplo de como se realiza una convolución. En este caso, los pesos de esta capa están formados por una matriz de 3x3 dimensiones. Realizar una convolución sobre la matriz de 7x7 de la figura 2.9 correspondería a multiplicar cada matriz de 3x3 contenida dentro de la matriz 7x7 por los pesos de forma ordenada.

A continuación, en la ecuación 2.4 se muestra un ejemplo del operador de convolución entre dos matrices: M y W. La primera correspondería a la entrada a la capa convolucional y la segunda correspondería a los pesos de esa capa. Los valores seleccionados para cada matriz son arbitrarios. $\langle \cdot, \cdot \rangle_F$ representa el *producto escalar de Frobenius*.

$$
\begin{aligned}
M \otimes W &= \begin{bmatrix} 1 & 2 & 3 \\ 3 & 4 & 5 \\ 5 & 6 & 7 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} \left\langle \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right\rangle_F & \left\langle \begin{bmatrix} 2 & 3 \\ 4 & 5 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right\rangle_F \\ \left\langle \begin{bmatrix} 3 & 4 \\ 5 & 6 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right\rangle_F & \left\langle \begin{bmatrix} 4 & 5 \\ 6 & 7 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right\rangle_F \end{bmatrix} \\
&= \begin{bmatrix} 5 & 7 \\ 9 & 11 \end{bmatrix}
\end{aligned}
\tag{2.4}
$$

La característica más relevante que la diferencia de otro tipo de redes es que aprovecha la información espacial de los datos de entrada. Esto lo convierte una buena opción a la hora de trabajar con cualquier tipo de señal, como por ejemplo imá-

genes e incluso audio, aunque esta última es menos común. Esto sucede gracias a que los pesos no solo se aplican sobre cada celda individualmente (esto sucedería si se hiciera una convolución con una matriz de pesos de 1x1), si no que los pesos suelen considerar también el vecindario del píxel. El tamaño de estos pesos es regulable y, de hecho, varía a lo largo de las múltiples capas convolucionales de una red. En las capas iniciales de la red, los tamaños de ventana suelen ser más grandes, ya que los tamaños en la entrada suelen ser muy grandes y, por ejemplo en el caso de imágenes, un único píxel no almacena mucha información en una imagen de 256x256. Progresivamente, mientras decrece el tamaño de entrada a cada capa de esta, las ventanas se hacen más pequeñas, puesto que los nuevos valores que albergan los píxeles de la imagen albergan información de varios píxeles de la imagen de entrada o, en otros términos, tienen mayor valor semántico.

Cada capa de la red tiene vinculados una serie de pesos, los cuales son optimizados durante el entrenamiento de la red para reducir el error mediante *backpropagation*. Cada ventana está vinculada con la extracción de una característica de la capa anterior. En las primeras capas de la red, las características extraídas son genéricas ya que se suelen extraer, por ejemplo, los bordes de los objetos. Sin embargo, conforme se añaden capas las características que se extraen se vuelven más complejas y dependen más del contexto del problema sobre el cual se está entrenando [24]. Al minimizar el error mediante *backpropagation*, se ajustan los pesos para que sean más precisos a la hora de extraer las características relevantes para el problema. Por ejemplo, si la red se entrena para detectar caras, las primeras capas extraerán probablemente los bordes de la cara y las capas más profundas buscarán extraer características más complejas como por ejemplo el tamaño y forma de los ojos.

Es muy común dentro de las CNN's que al final de la red se incluya un clasificador *Multi Layer Perceptron (MLP)* para que este utilice las características que ha extraído la CNN y realice una clasificación o *regresión*. El uso de este clasificador al final de la red tiene la ventaja añadida de que también se entrena con *backpropagation* y por tanto puede propagar el gradiente del error a la CNN. Esta última capa nos devuelve una medida de cuan parecidas son las características extraídas por la última capa de la CNN con respecto a las características esperadas de una clase. Estas características esperadas de cada clase también se modifican durante el entrenamiento mediante *backpropagation*.

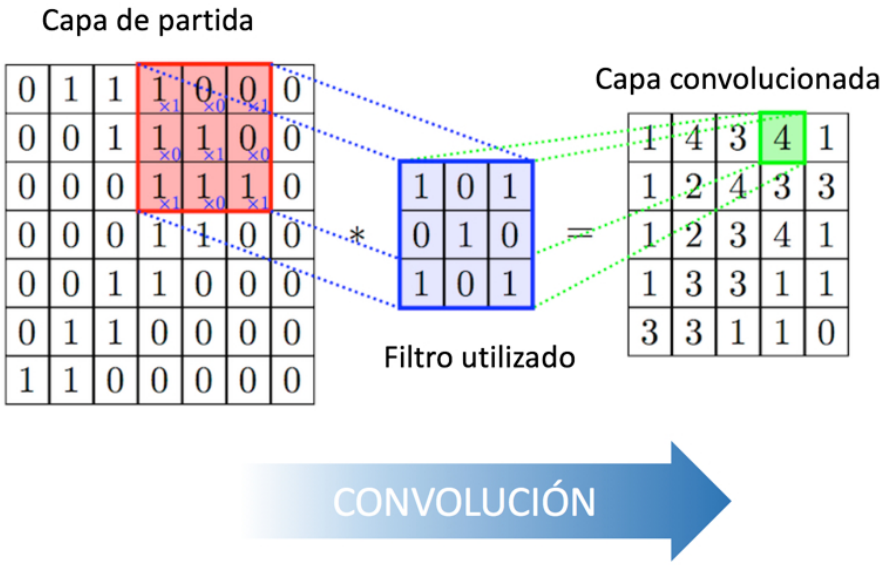


Figura 2.9: Operación de convolución que extrae las características de la imagen de entrada.

Multi-View CNN (MVCNN)

Existen múltiples formas de gestionar el entrenamiento con múltiples imágenes como, por ejemplo, concatenar las imágenes a lo largo del canal de color en la entrada de la red y procesarlas como si fuera una única imagen. De esta forma, la red aprenderá a extraer las características relevantes de cada imagen y a concatenarlas. Sin embargo, esto presenta varios problemas:

- El orden en el cual se ordenan las imágenes a la hora de concatenarlas a lo largo del canal de color es determinante. De utilizar esta clase de arquitectura se deberían tomar una serie de medidas:
  - Se debería entrenar la red para todas las combinaciones posibles entre las imágenes, lo cual aumenta con creces la complejidad del entrenamiento y su tiempo dependiendo del número de imágenes.
  - Se debería imponer un criterio de ordenación de las imágenes. En Zerogravity3D no existe tal criterio ya que cada cámara puede tomar una captura de cualquier perspectiva del objeto.
- El tamaño de la red escala con el número de imágenes.
- Se necesitaría entrenar la misma extracción de características genérica para cada imagen concatenada. Como se explica en el apartado anterior, las primeras capas de las redes convolucionales extraen características que son comunes para todo tipo de problemas e imágenes. En el caso de Zerogravity3D, si se concatenan varias imágenes a lo largo del canal de color signi-

ficaría que se deberían entrenar esta clase de extracciones de características un factor de 16 veces más.

Por todo esto, en este trabajo se utiliza otro procedimiento presentado en [21]. Consiste de una única red CNN tradicional que procesa las múltiples vistas del objeto por separado mostrando en su salida las diferentes características extraídas de las vistas. Es una única red que se entrena con cada imagen por separado. De esta forma, el tamaño de la red puede ser menor que si las imágenes se concatenaran, y además las características genéricas se entrenan solo una vez.

Conceptualmente, MVCNN permite utilizar cualquier arquitectura de red CNN, permitiendo incorporar a esta los avances en topología de las redes. En [21], el autor utiliza la red VGG [6], aunque en este trabajo se utiliza ResNet-18 [8], la cual es más reciente e, incluyendo las conexiones residuales entre capas, obtiene una mayor precisión. Ambas redes tienen al final una capa de clasificación (Multi Layer Perceptron, por ejemplo) que obtiene la clase a partir de las características extraídas.

Las características extraídas por la red para cada una de las vistas son combinadas para formar un único vector de características extraídas del modelo 3D. Por ejemplo, en [21] se concatenan extrayendo el valor máximo de cada característica de entre todas las vistas (element-wise maximum). Estas características del modelo 3D son posteriormente utilizadas por un clasificador para determinar la clase del modelo.

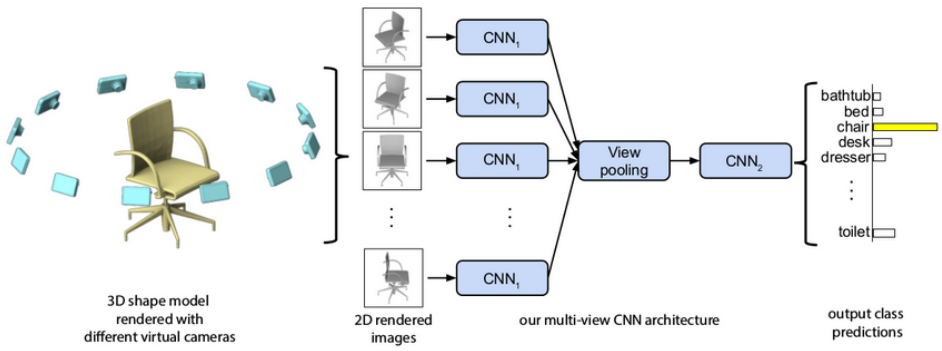


Figura 2.10: Estructura de una red MVCNN [21].

2.4 Clasificador con rechazo

Este clasificador, presentado en [2], determina la clase a la que pertenece el objeto basándose en las distancias euclídeas en el espacio de características. El hecho de que esté basado en distancias permite rechazar las muestras que no se ajusten a ninguna clase utilizando varios umbrales. Este clasificador está incluido dentro del sistema de Zerogravity3D y es utilizado en la experimentación para obtener los resultados finales para la comparativa entre los distintos extractores de características.

La estructura del clasificador divide la clasificación en dos fases bien diferenciadas. En la primera, la muestra que está siendo inspeccionada es procesada por un *K-Nearest Neighbours (KNN)*. Este determina cual es la clase a la cual pertenece basándose en las distancias euclídeas a sus K prototipos más cercanos, y selecciona la clase más cercana. Se calcula la cercanía mediante la suma de la inversa de la distancia a los K-vecinos que pertenecen a cada clase por separado (2.5).

$$S_a(x) = \hat{P}(c_i|x) = \frac{\sum_{j \in c_i} \frac{1}{d(x, y_j)}}{\sum_{j=1}^{K} \frac{1}{d(x, y_j)}}, f(x) \in [0, 1] \tag{2.5}$$

Al final de este proceso se determina la clase a la que pertenece la muestra o si es una muestra ambigua, es decir, que el clasificador está indeciso entre dos o más clases (para esto se selecciona un umbral para el valor de S_a llamado T_a), en cuyo caso habría que rechazarla.

El parámetro K se configura para cada experimento, y depende en gran medida de el número de prototipos de entrenamiento. Se han realizado numerosos estudios acerca de cómo ajustar este valor, y se establece que K debe ser menor o igual a $\sqrt{N_{prototipos}}$ [7][12][10][11]. Se utiliza el mismo valor de K para ambas fases del clasificador.

En la segunda fase se busca determinar si, dentro del contexto de la clase, este objeto es muy distinto a lo esperado, es decir, es un *outlier* que también habría que rechazar. Para esto se aísla la clase seleccionada en la fase anterior como si fuera la única en el espacio, es decir, se trabaja en modo "one-class classification". Después se construye un histograma formado por todas las distancias medias

entre cada *prototipo* de entrenamiento y sus vecinos más cercanos V dentro de la clase C (2.6).

$$g(x) = \frac{1}{N_y} \sum_{y \in C_x}^{V} d(x, y) \tag{2.6}$$

A los extremos del histograma se encuentran las muestras que más cerca y más lejos estén de sus vecinos. Estas muestras reciben el valor de $S_d = 0$ y $S_d = 1$, respectivamente. Los valores intermedios de S_d se calculan mediante la acumulación del histograma. Para una muestra de test, se calcula las distancias a sus vecinos más cercanos y se selecciona el S_d que corresponde a la barra que más se ajuste a la distancia calculada. Por supuesto, este valor también debe ser tener asociado un umbral para descartar las muestras que estén muy alejadas de sus vecinos. Un ejemplo de histograma de distancias para una clase se muestra en la figura 2.11.

Si se selecciona un umbral de S_d, llamado T_d, igual a 1 significa que solo pueden ser aceptadas piezas cuya distancia a sus vecinos sea como máximo la que tiene el *prototipo* más alejado de sus vecinos. Si queremos tener un $T_d > 1$ (como sucede para algunos experimentos) se utiliza la curva que describe el histograma acumulado. Se seleccionan dos puntos de la curva (por ejemplo, los valores de $S_d = 0{,}5$ y $S_d = 1$) y proyecta mediante interpolación lineal la recta que forman hasta el valor de T_d seleccionado. Cualquier muestra de test que supere la distancia media a sus vecinos que marca este punto, se rechaza por ser muy distinta.

A lo largo de este trabajo, las piezas clasificadas como desconocidas o ambiguas se engloban dentro de un mismo grupo. Este grupo se separa de las piezas clasificadas, ya sea dentro de la clase correcta o la clase incorrecta. Posteriormente, se necesitarán aplicar medidas sobre estas piezas puesto que han resultado ser demasiado distintas a lo esperado por el conjunto de entrenamiento. Entre las posibles medidas se incluyen:

- Una segunda inspección mediante Zerogravity3D, ya que puede haber sido descartada porque el lanzamiento de la pieza haya resultado en una extracción de características pobre.
- En última instancia, una inspección manual.

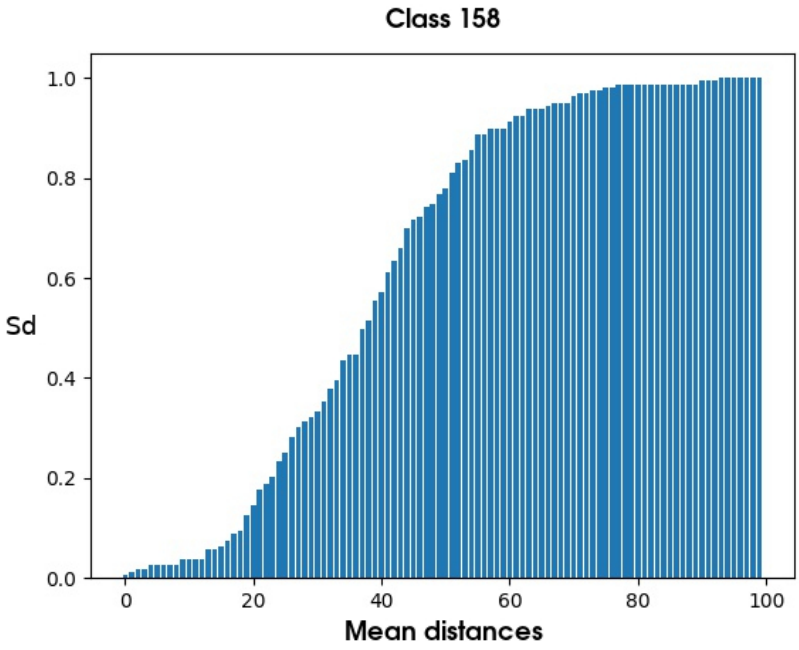


Figura 2.11: Histograma de distancias medias acumulado para la clase 158.

CAPÍTULO 3

Análisis del problema

3.1 Modelado conceptual

La experimentación necesaria para el proyecto sigue una estructura clara y ordenada, desde la obtención de las bases de datos hasta la evaluación de los resultados. La figura 3.1 muestra el proceso que siguen los datos desde que se obtienen los modelos 3D (en la sección 4.1 se explica como se obtienen), se extraen sus características correspondientes y finalmente se obtienen los resultados.

Los extractores de características de las vistas comparten el mismo proceso hasta el punto en el cual se construyen las bases de datos de entrenamiento, test y validación. Por el contrario, la experimentación con las características 3D se realiza al principio, poco después de aplicar una serie de deformaciones a los modelos 3D. Después de cada nodo, se almacenan los resultados en un fichero serializable para reducir los tiempos de experimentación. Todos estos apartados se exponen y se explican en la sección 4.

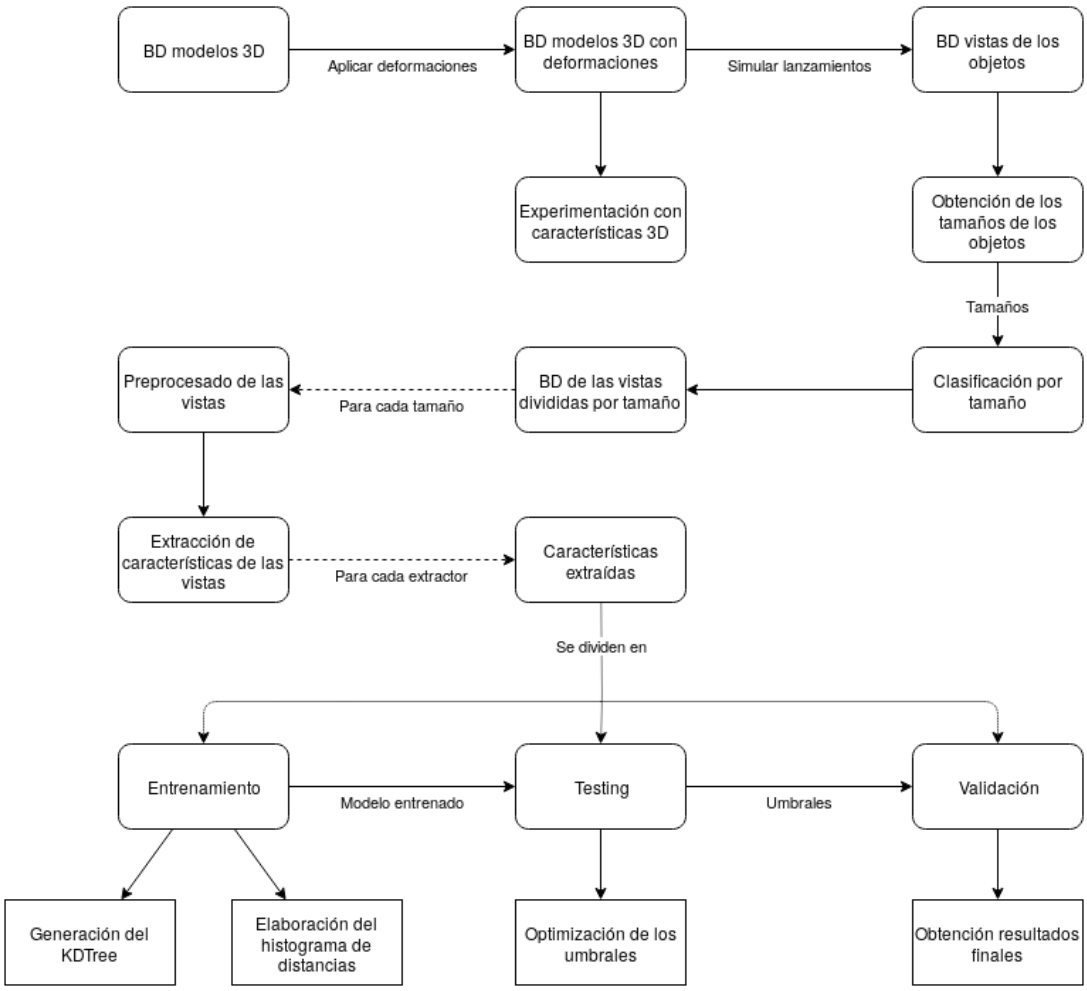


Figura 3.1: Diagrama de flujo completo del proceso de experimentación.

CAPÍTULO 4

Experimentación

4.1 Base de datos

Con el fin de realizar la comparativa, es necesario compilar una base de datos significativa y que sea relevante para el contexto de la inspección industrial. Por tanto es necesario seleccionar una base de datos que contenga multitud de piezas industriales distintas para simular el comportamiento de una fábrica donde Zerogravity3D pueda estar instalado. Se selecciona una página web[1] que contiene multitud de piezas industriales de diferentes categorías en forma de modelos 3D. Se obtienen 225 clases distintas, entre las cuales se pueden distinguir 6 grupos: tornillos, rodamientos, arandelas, muelles, coronas y casquillos. En la tabla 4.1 se muestra la cantidad de clases que hay por cada grupo.

Superclases	**Nº Clases**
Arandelas	51
Tornillos	94
Casquillos	18
Coronas	5
Muelles	31
Rodamiento	26
Total	225

Tabla 4.1: Número de clases por cada grupo de piezas.

Estos 225 modelos constituyen los modelos 3D base (uno por cada clase). Sobre estos se realiza una simulación de su lanzamiento dentro del sistema de Zerogravity3D y se obtienen 16 vistas por cada objeto. Estas 16 vistas forman un nuevo modelo 3D, lo que se denomina una reconstrucción del modelo.

[1] *www.mootio.com*

Como ocurriría en un caso real, existe una variación en la posición en que las cámaras capturan el objeto en cada lanzamiento, lo que permite que de un mismo modelo obtengamos conjuntos de 16 vistas distintos. Esta variación sucede durante la reconstrucción a 3D ya que suelen generarse abultamientos en planos cuando las vistas no se alinean correctamente con este. Son de estas variaciones del sistema de donde se obtiene la variabilidad suficiente para generar muestras distintas para entrenamiento, test y validación.

Sin embargo, si bien existen estas variaciones, es cierto que se pierde variabilidad intra-clase que puede surgir en piezas reales debido a alguna deformación durante el proceso de fabricación. Para simular este efecto se ha desarrollado un algoritmo capaz de deformar, con una intensidad regulable, las piezas originales.

4.2 Deformaciones en las piezas

Para deformar una pieza se parte de su modelo 3D, esto es, de su nube de puntos. El algoritmo desarrollado es ad-hoc y, aunque algunas deformaciones generadas pueden llegar a ser poco realistas, sirven para el propósito de generar modelos más variados.

En primer lugar, se seleccionan al azar un número determinado de puntos de la nube. Sobre estos puntos se aplica una fuerza aleatoria en el sentido determinado por la suma de las normales de los planos en los que participan. Se propaga ese desplazamiento al resto de puntos de la nube dependiendo de la distancia euclídea al punto sobre el cual se ha aplicado la fuerza en primer lugar. Se puede regular la fuerza aplicada sobre las piezas y también el número de fuerzas que actúan para ajustar el nivel de deformación que se busca.

Se han generado dos tipos de deformaciones distintas: unas deformaciones escogiendo 4 puntos distintos y desplazándolos dependiendo del tamaño de la pieza, y otras realizadas a mano (por ejemplo, simulando perdida de material por corte), realizadas en el trabajo [5], utilizando software para la edición de modelos 3D. La imagen 4.1 muestra las deformaciones que aparecen en los modelos 3D. Se generan 5 modelos 3D deformados automáticamente adicionales para cada clase, dando un total de 1417 modelos (225 modelos sin deformar, 1125 modelos deformados automáticamente y 67 modelos deformados a mano).

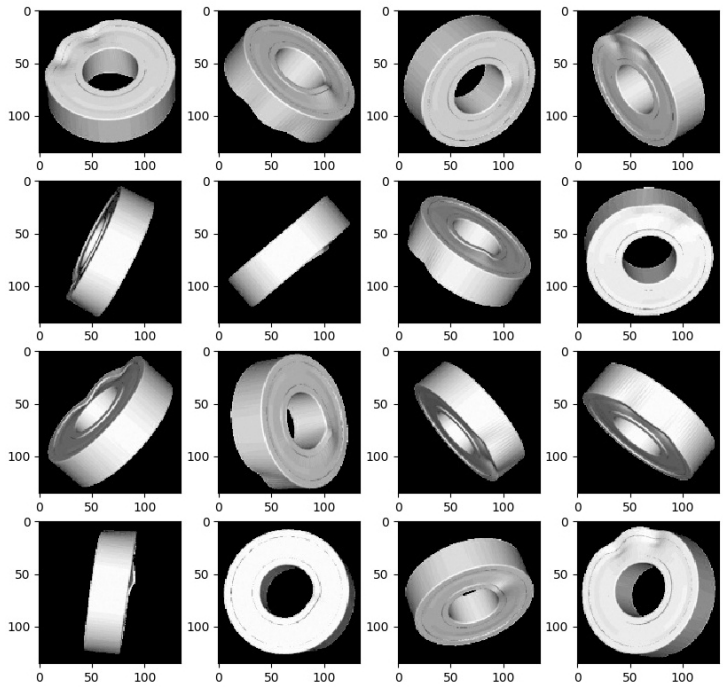


Figura 4.1: Modelo deformado.

Estas piezas deformadas se procesan como si se tratara del modelo original, es decir, se simulan lanzamientos de las mismas y se generan sus reconstrucciones. Estas son incluidas dentro de los experimentos como un complemento a las bases de datos. Esto significa que las piezas deformadas se incorporan en fases posteriores de la experimentación como una dificultad añadida a la base de datos de test.

Para los experimentos se generan dos bases de datos distintas. Una primera, sin piezas deformadas, que contiene 10 reconstrucciones por cada pieza para entrenamiento, 10 piezas más para validación y 10 para test. Además se genera otra base de datos, más compleja, que contiene piezas deformadas. Utilizando las mismas muestras de entrenamiento, se generan 5 reconstrucciones para validación y test de todas las piezas, tanto no deformadas como deformadas. La cantidad de reconstrucciones de este último apartado se puede observar con mayor claridad en la tabla 4.2. Subir más el número de muestras de entrenamiento implicaría un overtraining del clasificador. Es crucial que el clasificador desarrollado sea capaz de generalizar su conocimiento a muestras que no hayan sido incluidas en la base de datos de entrenamiento. Un problema común de los clasificadores es que a mayor número de muestras, y sin ninguna clase de penalización, tienden a sobreentrenar, es decir, memorizar las muestras de entrenamiento y no ser capaces de generalizar a partir de ellas.

Origen	Nº clases	Nº reconstrucciones totales
Clases originales	225	1125
Clases deformadas	1125	5625
Clases deformadas a mano	67	335
Total	1417	7085

Tabla 4.2: Número de reconstrucciones por cada tipo de pieza.

4.3 Extracción de características de la reconstrucción 3D del objeto

Actualmente el sistema de clasificación de Zerogravity3D utiliza este tipo de extracción de características. Para cada pieza entrante, obtiene 16 vistas desde distintos ángulos y reconstruye la pieza para generar un modelo 3D. De este modelo 3D se extraen 3 características geométricas distintas: el área, el volumen y la desviación típica de los puntos.

Es necesario medir el funcionamiento de este clasificador sobre la base de datos seleccionada. Se utilizan 10 reconstrucciones para entrenamiento, 10 para testing y 10 para validación. No se utilizan modelos deformados. Los resultados de este experimento se muestran en la tabla 4.3.

	Resultados
Ok	1981/2260 (87.655 %)
Mismatch	31/2260 (1.372 %)
Rejection	248/2260 (10.973 %)

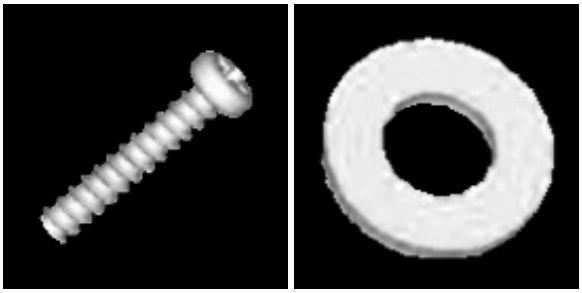
Tabla 4.3: Resultados para los 10 lanzamientos de validación utilizando el área, volumen y desviación típica. Parámetros: K = 3, T_a = 0.8 y T_d = 1.1.

Estas características extraídas son bastante genéricas y pueden resultar poco discriminativas, aunque obtienen una tasa de acierto alta para piezas no deformadas. Sin embargo, como se muestra en la tabla 4.4, no funcionan bien cuando existe una deformación en las piezas. Se propone añadir a estas características el histograma de distancias entre los puntos del modelo 3D y su centroide [14]. Un ejemplo de estos histogramas se puede visualizar en la figura 4.2.

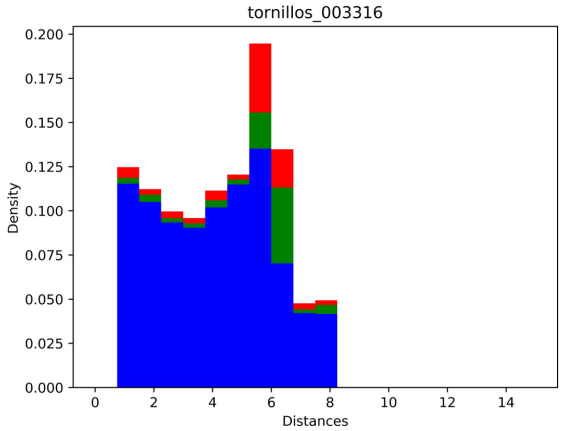
Cada característica se encuentra normalizada para acotar el rango. En el caso de uso de Zerogravity3D se conoce el tamaño máximo de las piezas que van a poder ser utilizadas dentro del sistema. Por esto se puede definir una esfera que acote

Resultado	Todas	Solo deformadas
Ok	4419/7115 (62.108 %)	3487/5985 (58.262 %)
Mismatch	349/7115 (4.905 %)	340/5985 (5.681 %)
Unknown	807/7115 (11.342 %)	803/5985 (13.417 %)
Ambiguous	1540/7115 (21.644 %)	1355/5985 (22.64 %)

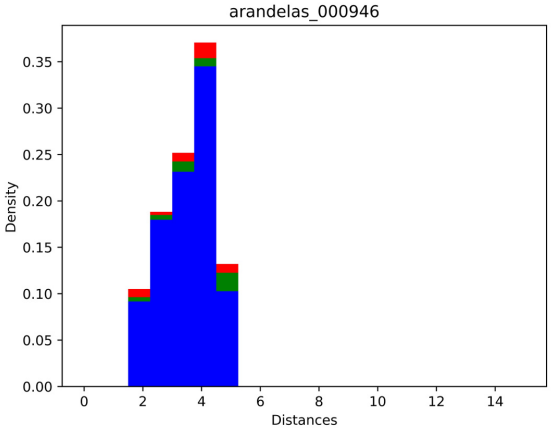
Tabla 4.4: Resultados del clasificador sin histogramas para las piezas deformadas. Parámetros: K = 5, T_a = 1 y T_d = 2.5



(a) Tornillo 003316 (b) Arandela 000946



(c) Histograma de distancias del tornillo 003316



(d) Histograma de distancias de la arandela 000946

Figura 4.2: Ejemplos de histograma de distancias. Se discretizan en 15 barras las distancias de todos los puntos al centroide de la pieza.

el tamaño máximo de las piezas que van a ser lanzadas. Se utiliza en volumen, el área y la desviación típica de los puntos en esta esfera para normalizar las características de los modelos 3D utilizados. En el caso de los valores del histograma de distancias, se normaliza de tal forma que la suma de los valores de todas las barras es 1.

Esta esfera también nos va a servir para generar el histograma de distancias. Se utiliza el radio de la esfera para dividir el espacio en 20 partes iguales (puede visualizarse como 20 esferas concéntricas de distinto tamaño) que representan las 20 barras que tiene el histograma. De esta forma el histograma almacena la probabilidad de que un punto del objeto se halle a una determinada distancia del centroide. Todas las características se concatenan para formar un único vector y, por tanto, cada modelo 3D se representa con un vector de 23 características.

El criterio de selección de esta nueva característica no es arbitrario. Como se explica en la sección 2.3, un requisito para las características que se extraigan es que deben ser invariantes a la rotación y la traslación, pero no invariantes al escalado ya que existen piezas que únicamente se diferencian por su tamaño. Todo esto lo cumple el histograma de distancias de los puntos del modelo 3D al centroide. Debido a que se considera el centroide como punto de referencia, todas las distancias calculadas serán invariantes a la traslación. Debido a que se selecciona una esfera máxima y se divide el espacio con respecto a esta, y no con respecto al tamaño del objeto, el histograma es variante al escalado. Y, por último, cualquier rotación de la pieza no modifica la distancia de cada punto al centroide del modelo 3D.

En la tabla 4.5 se puede observar la mejora en los resultados que aporta incluir el histograma de distancias a la lista de características extraídas. El algoritmo, aun con relativamente pocas muestras de entrenamiento, es capaz de alcanzar el 92.13 % de tasa de acierto de las piezas de validación. Esto supone una mejora frente al clasificador original en tanto que aumenta la tasa de acierto un 4.5 % y reduce la tasa de error un 1.2 %.

Como experimento final, cabe medir la tasa de acierto y error para la base de datos con las piezas deformadas, para ser comparable con los otros extractores considerados. La tabla 4.6 muestra estos resultados finales. Una vez se añaden las piezas deformadas, estas suponen una variación tan grande sobre las muestras originales que provocan la mayoría de los errores de clasificación; además de

Resultado	**Con histograma**	**Sin histograma**
Aciertos	2073/2250 (92.133 %)	1971/2250 (87.6 %)
Errores	4/2250 (0.178 %)	31/2250 (1.38 %)
Desconocidos	173/2250 (7.689 %)	248/2250 (11.02 %)

Tabla 4.5: Resultados utilizando reconstrucciones sin deformar. Parámetros: K = 5, T_a = 1 y T_d = 2.5

la mayor parte de piezas desconocidas.

Este extractor de características tarda 2 segundos en calcular las características para cada pieza, siendo la parte más costosa de esta fase la necesaria reconstrucción 3D del objeto.

Resultado	Todas	Solo deformadas
Aciertos	5019/7085 (70.84 %)	3984/5960 (66.846 %)
Errores	287/7085 (4.05 %)	284/5960 (4.765 %)
Desconocidos	1779/7085 (25.11 %)	1692/5960 (28.389 %)

Tabla 4.6: Resultados utilizando reconstrucciones deformadas. Parámetros: K = 5, T_a = 1 y T_d = 2.5

4.4 Separabilidad por tamaño

Las piezas de distintas clases tienen tamaños considerablemente variados. Esto da lugar a una separabilidad entre las clases previa a cualquier entrenamiento más elaborado utilizando la información de las imágenes. Por tanto, se pueden preprocesar las imágenes para que no se comparen clases que tengan tamaños dispares.

Una posibilidad para clasificar las imágenes por tamaño es incluir este como una característica más al vector de características de cada pieza. Sin embargo, incluir el tamaño dentro de este vector implica la necesidad de escalarlo de tal forma que tenga la magnitud necesaria para tener impacto dentro de la métrica de distancia. Cuanto mayor es el tamaño del vector de características, menor es el impacto que tiene una nueva característica adicional. Por esto se propone utilizar una suerte de árbol binario.

Primero, se obtienen todos los tamaños en píxeles para cada una de las vistas de las reconstrucciones. De entre todos los tamaños de las vistas seleccionamos los

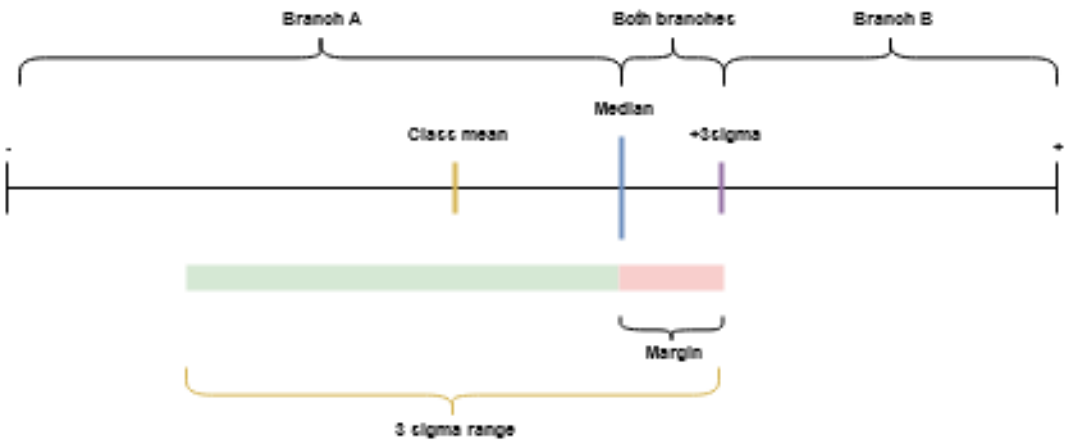


Figura 4.3: División a nivel de nodo de los tamaños de las piezas. El valor del margen lo establece la clase que más atraviese la mediana, es decir, el margen máximo necesario para que todas las piezas caigan en le misma rama, en este caso la rama A. El valor de la mediana no es la mediana de la clase, sino la mediana de todos los tamaños que hayan llegado hasta ese nodo, independientemente de la clase.

valores de ancho y alto máximos, y luego nos quedamos con el valor más alto de los dos. Con este tamaño se entrena el árbol binario de búsqueda, en el cual cada nodo del árbol extrae la mediana de los tamaños que le llegan. Este valor no es suficiente ya que pueden existir clases cuyas piezas estén a ambos lados de la mediana. Por ello se estima un margen utilizando las clases cuyo 3σ atraviese la mediana. Mediante la mediana y el margen se predice si el tamaño para ese nodo es pequeño, grande o si el tamaño es ambiguo y se deben explorar ambas ramas. Un ejemplo de esto se muestra en la figura 4.3. Prediciendo si el tamaño es ambiguo o no nos permite reducir la probabilidad de que una pieza correcta con un tamaño ligeramente distinto a lo existente en la base de datos de entrenamiento explore ramas en las cuales no se encuentre ninguna muestra de su clase y no consiga clasificarse correctamente.

Para cada una de las hojas del árbol se entrena un clasificador con rechazo con las muestras de entrenamiento que hayan llegado hasta ese nodo. Se configura el árbol para que tenga una profundidad de 3, es decir, que tenga 8 hojas.

El rango de los tamaños en píxeles de las piezas es de [46-808]. La distribución de los tamaños se muestra en el histograma de la figura 4.4. Tras aplicar el clasificador por tamaño, los rangos de tamaños que tiene cada hoja del árbol son: [46-136], [126-162], [148-193], [165-217], [190-268], [208-328], [282-413] y [351-805].

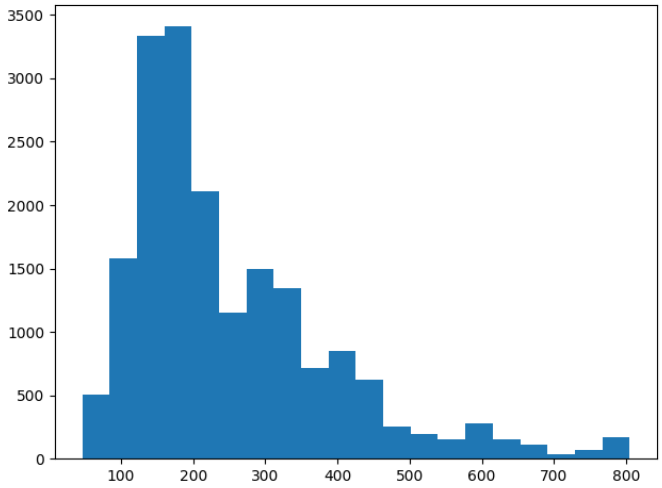


Figura 4.4: Histograma de tamaños de las 225 clases de piezas.

4.5 Preprocesado de las vistas de los modelos 3D

Las vistas obtenidas de las cámaras son computacionalmente ineficientes de procesar directamente mediante los extractores de características. Por tanto, es necesario modificarlas de forma que se minimice el tiempo de cómputo, el consumo de memoria y que se maximice la calidad de las características extraídas. Gran parte de las técnicas de preprocesado que se incluyen no pueden ser aplicadas a las capturas si se pretende reconstruir posteriormente; estas técnicas se aplican únicamente dentro de la experimentación con extractores de características de las vistas.

Como se puede visualizar en la figura 4.5, el preprocesado de las imágenes consta de tres etapas:

- Segmentado.
- Padding.
- Escalado.

Segmentado

Consiste en extraer la zona de interés (ROI) de una imagen. Debido a que los objetos ocupan una pequeña área dentro de las capturas que realizan las capturas de Zerogravity3D, es desaconsejable trabajar con el tamaño completo de las imágenes. De esta forma los algoritmos posteriores pueden trabajar con imágenes de menor tamaño y así reducir el coste computacional y el espacio en memoria y disco.

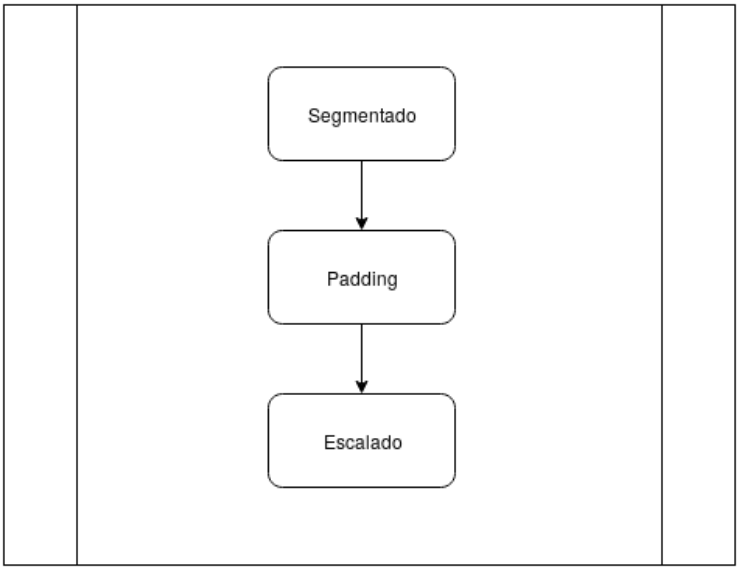


Figura 4.5: Etapas del preprocesado de las imágenes previo a la experimentación.

El segmentado se realiza conociendo el color de fondo de las imágenes. Este se sustrae de todos los píxeles de la imagen. En la figura 4.6, el color de fondo es el rojo. Al remover este color de todos los píxeles, aquellos que contienen un valor que supere un cierto umbral mayor que cero se considera que contienen el objeto. El motivo del umbral es que, debido a que la compresión JPG introduce ruido en las imágenes, es posible que cuando se sustraiga el color rojo de cada píxel, el color del fondo no se convierta en un color negro puro ([0, 0, 0]) sino que tenga una serie de oscilaciones. Esto sucede en mayor medida en los píxeles cercanos al objeto.

Se construye una máscara mediante todos los píxeles que superen el umbral y se recorta para que el objeto ocupe toda la imagen. En este punto, el color de fondo de la imagen debe de ser negro (debido a la sustracción del paso anterior) y el objeto debe de ocupar toda la imagen, como se muestra en la figura 4.7.

Padding y escalado

Por simplicidad, se rellenan los bordes de la imagen con negro para generar una imagen cuadrada. De esta forma podemos establecer un tamaño común para todos los tipos de pieza, independientemente de su forma. Posteriormente todas las imágenes son escaladas al mismo tamaño, siendo este imágenes cuadradas de 135x135.

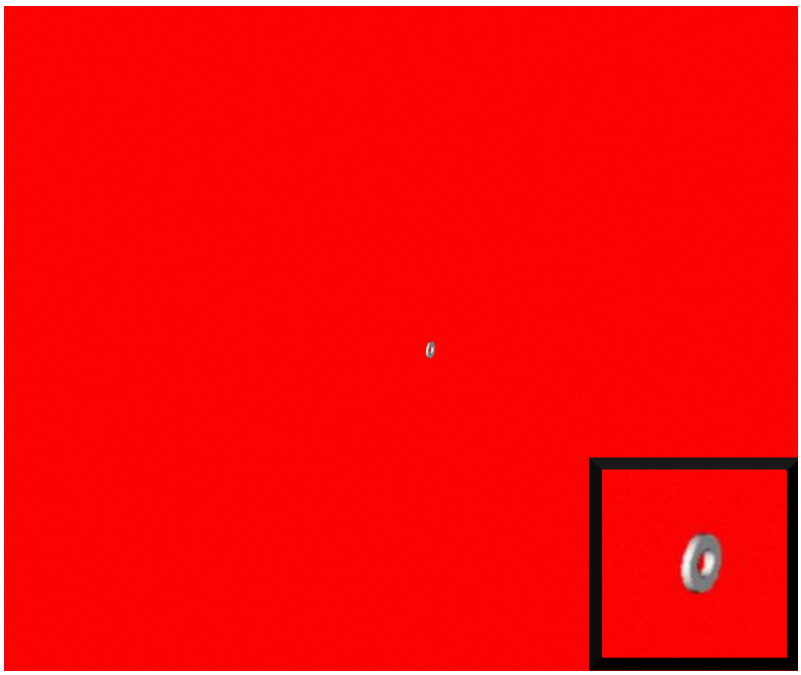


Figura 4.6: Ejemplo de captura de las cámaras de Zerogravity3D. Abajo a la derecha se muestra el objeto ampliado.

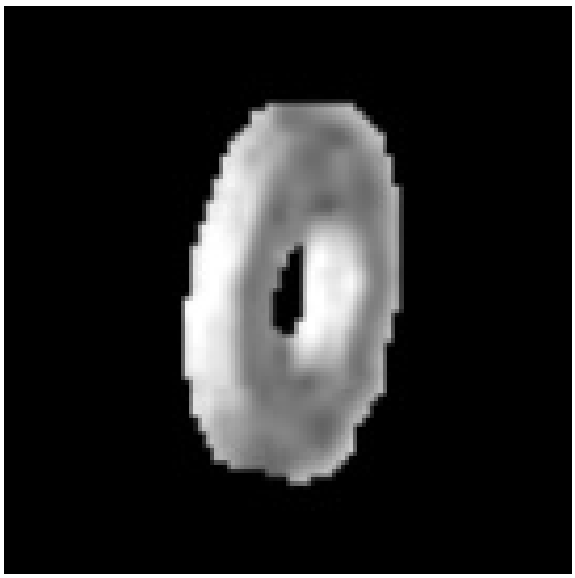


Figura 4.7: Vista del objeto tras el segmentado.

4.6 Extracción de características de las vistas del objeto

En este apartado se dejan atrás los modelos tridimensionales reconstruidos para utilizar únicamente sus vistas. Aquí se implementan las técnicas presentadas en la sección 2 para la extracción de características de imágenes.

4.6.1. Técnicas clásicas

Tanto RLBP como Zernike generan un vector de tamaño constante para cada una de las vistas de la pieza y, por tanto, cada pieza tiene el mismo número de características que la representan (16 vectores de características). Cada una de las vistas vota cuál es la clase a la que cree que pertenece y la clase mayoritaria, si pasa un cierto umbral, es a la que corresponde la pieza. Se comparan las dos clases con más votos, C_1 y C_2, para calcular este valor de confianza S_r sobre el que aplicaremos el umbral. Se calcula mediante la formula 4.1.

$$S_r = \frac{C_1}{C_2} \tag{4.1}$$

Este umbral será referido como T_r. Al ser procesada cada vista por separado, estas pueden ser consideradas individualmente como ambiguas o desconocidas (aplicando los respectivos umbrales T_a y T_d), y por tanto ser rechazadas para la clasificación.

Para las pruebas que se muestran a continuación se establecen inicialmente unos umbrales muy laxos para cada uno de las variables del problema: se aceptan como válidas reconstrucciones que tengan un $T_r > 1$, es decir, que como mínimo no empaten en votos las dos clases más votadas. Los otros umbrales, T_a y T_d, se configuran para aceptar cualquier vista ambigua y que esté dentro del rango de distancias a sus k-vecinas del conjunto de entrenamiento, es decir, $T_a = 0$ y $T_d = 1$.

A partir de estos umbrales iniciales, se optimizan los resultados explorando el vecindario de los umbrales establecidos con el fin de maximizar una función de puntuación. Esta función está definida tal que:

$$score(aciertos\,\%, 100 - fallos\,\%) = (100 + aciertos - fallos)/2 \tag{4.2}$$

RLBP

A efectos prácticos, no todas las 59 barras del histograma que genera RLBP son relevantes. Debido a que se ordenan los patrones para que el primer dígito pertenezca al píxel que mayor gradiente tenía con respecto al central, es imposible que existan patrones con una cierta configuración. Si el primer dígito es un 0, implica que todos los demás deben ser también 0, ya que de haber algún 1 debería estar en la primera posición tras el ordenado. Esto implica que se puede reducir el tamaño del vector de 59 a 31 dimensiones.

Se han realizado pruebas para comprobar qué K (número de vecinos) es la óptima para el problema. Se comprueban los resultados obtenidos para varias configuraciones de este valor: 3, 5, 11, 15 y 63. Estos valores han sido seleccionados al azar, aunque utilizando el criterio descrito en la sección 2.4, que establece que el valor de K debe de ser como máximo $\sqrt{N_{prototipos}}$. Los resultados se muestran en la tabla 4.7. Se establece para todos los extractores de características que el valor de K óptimo, según el número de muestras de entrenamiento, es $K = 5$.

Valores de K	Precisión
K = 3	70 %
K = 5	70.93 %
K = 11	69.45 %
K = 15	68.7 %
K = 63	63 %

Tabla 4.7: Resultados obtenidos para la comparativa entre distintos valores de K.

Se entrena el clasificador con rechazo con la misma cantidad de reconstrucciones que en la Sección 4.3 para cada base de datos. Los resultados se muestran en la tabla 4.8. El tiempo de extracción de características por cada vista es de 10 ms

	Número de reconstrucciones de test	Tras optimizar
Acertados	5007 (70.93 %)	4767 (67.53 %)
Errores	1609 (22.79 %)	1038 (14.7 %)
Descartados	443 (6.28 %)	1254 (17.76 %)
Score	74.37	76.42

Tabla 4.8: Resultados para RLBP.

Zernike

El siguiente en la lista de experimentos es la extracción de características mediante los momentos de Zernike. Como se ha mencionado en la Sección **??**, se puede

ajustar el número de momentos de Zernike a extraer y con esto regular la precisión con la que se puede reconstruir la imagen original. Se puso el ejemplo con 49 y 256 momentos, pero para la experimentación se utilizan únicamente 49 para intentar minimizar la dimensionalidad del problema. Aun con esto se consigue una precisión superior a RLBP, tal y como se muestra en la tabla 4.9. Se utiliza el mismo valor para K. El tiempo de extracción de características por cada vista es de 125 ms.

	Número de reconstrucciones de test	Tras optimizar
Acertados	6302 (88.95 %)	6363 (89.81 %)
Errores	689 (9.72 %)	491 (6.93 %)
Descartados	94 (1.33 %)	231 (3.26 %)
Score	89.62	91.44

Tabla 4.9: Resultados utilizando Zernike.

Como experimentación adicional se ha probado a concatenar, para cada pieza, las características de sus vistas de la misma forma que se hace en MVCNN, es decir, extrayendo el máximo para cada característica. El número de muestras de entrenamiento se reduce drásticamente (una dieciseisava parte), lo que provoca que algunos clasificadores únicamente tengan un *prototipo* para algunas clases. Al solo haber un *prototipo* para algunas clases, cualquier muestra de test que pertenezca a esa clase será clasificada como ambigua. El número de aciertos se reduce de 88.81 % a 73.22 %. Conviene para una futura experimentación probar con más muestras de entrenamiento.

RLBP + Zernike

Como una experimentación adicional se propone combinar las características extraídas por Zernike con las características de RLBP. De esta forma se busca comprobar si la separabilidad entre las clases es mayor cuando la información de ambos extractores es puesta en común. Tras la concatenación obtenemos un vector de 80 dimensiones para cada vista. Como se muestra en la tabla 4.10, la combinación de las características obtiene un peor resultado que el uso de Zernike aislado y un resultado comparable a únicamente utilizar RLBP.

4.6.2. MVCNN

En este apartado se extraen las características de las vistas mediante redes convolucionales, en concreto, la ya presentada MVCNN. El proceso de entrenamiento

	Número de reconstrucciones de test	Tras optimizar
Acertados	5040 (71.14 %)	4908 (69.27 %)
Errores	1657 (23.39 %)	1147 (16.19 %)
Descartados	388 (5.48 %)	1030 (14.54 %)
Score	73,88	76.54

Tabla 4.10: Resultados utilizando Zernike y RLBP.

es similar al realizado en el trabajo [5], sin embargo el número de muestras es distinto para poder ser comparable con los otros experimentos.

Se utilizan técnicas para aumentar la base de datos como la rotación de las imágenes, puesto que, como se ha mencionado en el apartado *"4.3. Extracción de características de la reconstrucción 3D del objeto"*, las características finales extraídas deben ser invariantes a la rotación. Esto es necesario ya que, a diferencia de Zernike y RLBP, las CNNs no son invariantes a la rotación, únicamente a la traslación. Se fuerza esta invarianza mediante el entrenamiento con versiones de las imágenes rotadas.

Como base de datos de entrenamiento utilizamos 10 lanzamientos distintos de cada tipo de pieza y, a cada una, se le aplican 12 rotaciones desde 0º hasta 330º. Las bases de datos de test y validación están formadas, como en apartados anteriores, por 5 lanzamientos. Esto suma un total de 2250 objetos sin deformar para entrenamiento, 7085 para test y validación. El tiempo de extracción de las características es de 400 ms.

Entrenamiento

Para el entrenamiento se utiliza al final de la extracción de características una capa Perceptrón para propagar el gradiente del error al resto de la red. Como optimizador de la red se utiliza *Adam*, con un *learning rate* inicial de 0.001, el cual se disminuye a una décima parte cada vez que converge el entrenamiento, hasta un limite de 2 veces. No se aplica *weight decay*.

Con todo esto, tras 80 épocas de entrenamiento (unas 5 horas de entrenamiento en una NVIDIA RTX 2070) alcanzamos una precisión en la base de datos de validación de 95.57 %.

Esta es una tasa de acierto alta, sin embargo nos interesa también poder discriminar piezas que estén muy deformadas con respecto a la forma esperada de la

pieza. Se puede aplicar un umbral para descartar las confianzas de clase que devuelve el clasificador Perceptrón que no superen un valor, como se realizó en [5]. Sin embargo utilizaremos la aproximación del clasificador con rechazo.

Clasificador con rechazo

De forma similar a como se ha realizado en anteriores experimentos, se configuran los umbrales para ser muy laxos. En este apartado no existe el umbral T_r, presentado en la sección 4.6.1, ya que se juntan todas las características de cada vista en una única, la cual determina la clase a la que pertenece. Se obtienen los siguientes resultados:

	Número de reconstrucciones de test	Tras optimizar
Acertados	6712 (94.74 %)	6632 (93.61 %)
Errores	373 (5.26 %)	286 (4.04 %)
Descartados	0(0.00 %)	167(2.36 %)
Score	94.74	94.79

Tabla 4.11: Resultados de utilizar la extracción de características con MVCNN y el clasificador con rechazo.

CAPÍTULO 5

Comparativa

En lo referente a las características 3D, las nuevas características propuestas aportan una mejora frente a las existentes de un 8.7 % en precisión y reducen la tasa de error un 0.86 %. Esto permite tener las tasas de error más bajas frente a cualquier extractor de características.

En lo referente a extracción de características de imagen con técnicas clásicas, Zernike es el extractor de características que mejores resultados obtiene, por encima de RLBP y su combinación con este. Este incluso supera a la extracción de características del modelo 3D. Esto implica que es posible obtener mejores resultados, permite eliminar la etapa de reconstrucción 3D para la clasificación con rechazo y reducir el tamaño de los paquetes que envían las cámaras. Actualmente, las cámaras en Zerogravity3D envían la imagen completa a un nodo centralizado. Sin embargo, el hecho de poder paralelizar la extracción de características permite a las cámaras enviar únicamente el vector de características en vez de la imagen completa reduciendo, por tanto, el ancho de banda necesario. Se pasaría de enviar paquetes de 1 MB a paquetes de 1.53 KB.

Sin embargo, cuando se añade *Deep Learning* a la comparativa, MVCNN obtiene mejores resultados que cualquier otro extractor de características probado. Aun así, Zernike es capaz de obtener resultados similares aunque con un mayor porcentaje de error. Los resultados de la comparativa se muestran en la figura 5.1.

Evitar la reconstrucción 3D aporta ventajas adicionales. A la hora de reconstruir un objeto, el número de cámaras y su posicionamiento influye drásticamente en el resultado de la reconstrucción, por lo que si alguna cámara no funcionara correctamente, el sistema podría perder calidad en la clasificación. Sin embargo,

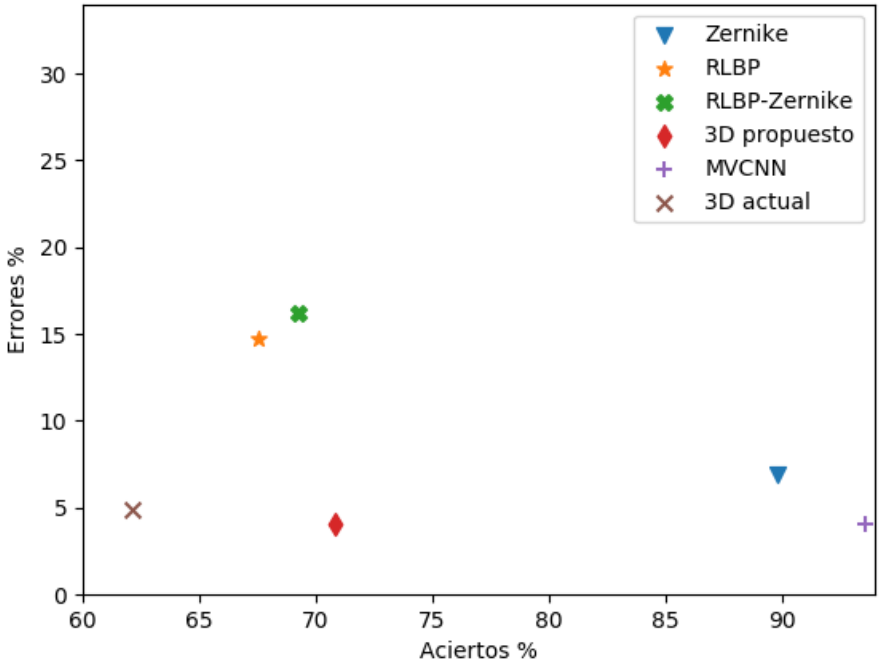


Figura 5.1: Comparativa de los extractores de características. La zona óptima es la esquina inferior derecha.

gracias al diseño actual del clasificador y los extractores de características de las vistas de los objetos, estos podrían seguir funcionando independientemente, y sin necesidad de volver a entrenar, al número de cámaras. Esto supone una ventaja añadida ya que lo hace robusto a esta clase de fallos.

Además, el uso de las vistas sin necesidad de reconstruir a 3D elimina la necesidad del calibrado de los parámetros extrínsecos (ver sección 2.1) de las cámaras debido a posibles movimientos que sucedan durante el funcionamiento. Esto se debe al requisito que se ha establecido de invarianza ante la traslación de las características, tal y como se expone en la sección 2.3.

En cuanto a tiempo de cómputo, RLBP es la técnica más rápida (10 ms) ya que es un algoritmo moderadamente simple y altamente paralelizable. La extracción de características mediante Zernike tiene una complejidad superior y por tanto es más costoso (120 ms). MVCNN, debido a la profundidad de la red, es más lenta que Zernike, costando 400 ms. La técnica más lenta es la extracción de características 3D, ya que la reconstrucción a partir de las vistas y la extracción de características geométricas tarda 2 segundos de media. En la tabla 5.1 se muestra

una comparativa entre los distintos algoritmos.

	Tasa de acierto	Tasa de error	Tiempo de cómputo
MVCNN	93.6 %	4 %	400 ms
Zernike	89.8 %	6.9 %	120 ms
RLBP	67.53 %	14.7 %	10 ms
Zernike + RLBP	69.3 %	16.2 %	130 ms
3D propuestas	70.84 %	4.05 %	2000 ms
3D existentes	62.1 %	4.91 %	1900 ms

Tabla 5.1: Comparativa entre todos los extractores de características con tiempos.

Frente a las características existentes, se puede observar que mediante MVCNN se obtiene una mejora de un 31.5 % en la precisión y una reducción de 0.91 % en la tasa de error. Además, se ahorran 1500 ms en el cómputo de las características.

En el caso de Zernike, se obtiene un aumento de 27.7 % en la precisión y un 1.99 % en la tasa de error. En este caso, la reducción de los costes computacionales es mayor, permitiendo ahorrar 1780 ms en la extracción. A continuación, únicamente conservamos estos dos extractores de características por ser los dos que mejores resultados obtienen y descartamos los restantes.

CAPÍTULO 6

Análisis de riesgos

En este apartado se resaltan los riesgos que implica la implementación de este proyecto dentro del sistema de Zerogravity3D y su futura puesta en producción. En primera instancia se enumeran los interesados en el proyecto (los stakeholders), ya que son los afectados por los riesgos en caso de acontecer. Posteriormente, se realiza un análisis exhaustivo de los riesgos:

- Se describen los riesgos a partir de sus causas, los eventos que lo provocan y sus efectos.
- Se evalúa su impacto de estos sobre los stakeholders, la probabilidad de que ocurran y se clasifica según su gravedad.
- Se elabora un plan de contingencia para cada riesgo.

6.1 Stakeholders

Los stakeholders principales de este proyecto son:

- Las empresas asociadas al Instituto Tecnológico de Informática.
- Los clientes que incorporen Zerogravity3D.
- El propio Instituto Tecnológico de Informática.
- El departamento de Investigación y Desarrollo.
- La Generalitat Valenciana.

Las expectativas de los stakeholders es variada pero en ocasiones compartida entre varios:

- El Instituto Tecnológico de Informática y el departamento de Investigación y Desarrollo esperan un producto capaz de generar valor para la empresa.

- El departamento de Investigación y Desarrollo y la Generalitat Valenciana esperan un producto que justifique las ayudas recibidas.
- Los clientes que incorporen Zerogravity3D y las empresas asociadas al Instituto esperan un sistema que mejore su productividad.

6.2 Riesgos

Se establece una lista de los riesgos de la implantación del producto mediante un compendio entre la experiencia previa y las lecciones aprendidas durante la experimentación. Las tablas 6.1, 6.2 y 6.3 muestran los riesgos que se han conseguido identificar y la tabla 6.4 muestra una posible oportunidad registrada.

Nombre	El entrenamiento de MVCNN no converje en producción.
Descripción	Debido a la naturaleza de las redes neuronales, se requiere un entrenamiento ad-hoc para cada problema sobre el que tienen que trabajar. Si se requiere que la red neuronal sea capaz de clasificar una pieza nueva, se deben incluir nuevas piezas de esta clase dentro de la base de datos y realizar una nueva fase de entrenamiento. Es importante encontrar una forma de determinar que esta fase de entrenamiento ha sido satisfactoria, de lo contrario los costes para la fábrica pueden ser inaceptables. Existen múltiples criterios para determinar si la red ha sido entrenada correctamente, aunque esto es más preciso dado un conjunto enorme de muestras de entrenamiento. En los casos reales, las fábricas no suelen contar con un conjunto etiqueta enorme de muestras ya que el etiquetado manual es costoso. Además, es posible también que la red, debido a su estructura, no tenga la capacidad suficiente para discriminar un número elevado de clases distintas con pocas muestras.

Causa	Debido a que Zerogravity3D es un prototipo y que la mayoría de la literatura realiza su experimentación con amplias bases de datos, no podemos predecir el funcionamiento de esta red neuronal en un entorno de producción real.
Evento	Se detecta en producción un escenario en el cual la red no es capaz de clasificar el conjunto de datos de entrenamiento debido a la falta de capacidad de la red.
Efecto	Se debe parar la máquina y sustituir el sistema de extracción de características.
Evaluación del riesgo	▪ Probabilidad: 35 %. Debido a la falta de experiencia en la puesta en producción de redes neuronales como sistema de extracción de características, es probable que surjan esta clase de eventos. ▪ Impacto: 100 %. Se debe de parar la línea de producción hasta que se encuentre la solución. ▪ Riesgo: 0.35 (Importante).

Plan de contingencia	▪ Intentar eliminar la causa: Se puede descartar el uso de MVCNN como extractor de características, a pesar de sus buenos resultados. ▪ Reducir el impacto del efecto: El sistema debe de estar preparado para cuando esto suceda. Se puede diseñar un interruptor lógico que, cuando sea necesario, cambie el extractor de características. Por ejemplo, se puede alternar entre el uso de las características que extrae MVCNN y las características de Zernike.

Tabla 6.1: Riesgo de que el entrenamiento de MVCNN no converja en producción.

Nombre	Aparición de piezas que el sistema no sea capaz de discriminar.
Descripción	La el tipo de piezas que pueden necesitar ser clasificadas es tan diverso que podrían necesitarse piezas que, por su forma, son muy parecidas a las piezas de otra clase. Esto provoca que las clases sean incapaces de ser discriminadas y se requeriría otro tipo de características, posiblemente no contempladas en este estudio. La base de datos es amplia y completa; se ha buscado una lo suficientemente exigente para tener más seguridad. Sin embargo, no está exenta de posibles limitaciones.
Causa	El origen del riesgo es parecido al que se muestra en la tabla 6.1. Zerogravity3D es un prototipo y no se ha probado que las características escogidas puedan funcionar sin ningún tipo de limitación en producción.

Evento	Se detecta en producción un escenario en el cual la red no es capaz de clasificar piezas que sean muy parecidas entre sí.
Efecto	Se debe parar la línea y realizar un nuevo estudio en busca de características más significativas.
Evaluación del riesgo	▪ Probabilidad: 30 %. Se ha utilizado una variada base de datos, aunque hay piezas que no han sido consideradas que pueden ser problemáticas. ▪ Impacto: 100 %. Implicaría la invalidez del trabajo realizado y se requeriría un tiempo adicional de experimentación cuando el sistema ya está en producción. ▪ Riesgo: 0.3 (Importante).
Plan de contingencia	▪ Intentar reducir la probabilidad del evento: Se puede establecer las piezas que van a ser procesadas por un cliente. Se puede realizar un estudio previo ad-hoc para asegurar al cliente que todas sus piezas son fácilmente diferenciables por nuestro sistema, además de asegurarle un índice propio de precisión. ▪ Reducir el impacto del efecto: Tener varios clasificadores preparados para poder examinar cual de ellos funciona mejor con el nuevo conjunto de piezas.

Tabla 6.2: Riesgo de que aparezcan piezas que el sistema no sea capaz de discriminar.

Nombre	Tiempos inasequibles a la hora de calcularlos en las placas vinculadas a cada cámara.

Descripción	Debido a que los tiempos a lo largo de este proyecto han sido calculados en un servidor con mayor capacidad de cómputo que el hardware final, es posible que los tiempos de cálculo de las características en estos sean demasiado grandes como para que sea una solución válida. Si bien no se estima que sea el caso, es posible que la tarea de extracción en combinación con otras resulte en este efecto. Si fuera este el caso, la productividad de Zerogravity3D se reduciría y, aunque mejore su precisión a la hora de clasificar, reduciría su valor ya que ralentizaría la línea de producción.
Causa	La reducción de capacidad de cómputo en los sistemas de procesamiento finales, es decir, las placas vinculadas a cada cámara.
Evento	Estimación de tiempos demasiado grandes durante las pruebas de implantación. Esto es, una vez se hayan implementado todas las optimizaciones posibles para el algoritmo y se comiencen las pruebas desde las placas, que el tiempo de predicción sea mayor que el existente.
Efecto	Se descarta la propuesta como inválida para el prototipo final.
Evaluación del riesgo	▪ Probabilidad: 25 %. Las placas del prototipo final propuestas tienen una cierta potencia de cómputo. ▪ Impacto: 50 %. Se podría seguir utilizando las características seleccionadas aunque deberán ser extraídas desde el servidor central. ▪ Riesgo: 0.125 (Tolerable).

Plan de contingencia	▪ Intentar eliminar la causa: Se pueden realizar pruebas en un entorno aislado con una sola placa. Se puede utilizar una simulación de una captura y medir los tiempos en este, sin necesidad de ser montada dentro el sistema de Zerogravity3D. ▪ Reducir la probabilidad del evento: Se pueden adquirir placas vinculadas a la cámara más costosas y con mayor capacidad de cómputo. Otra opción es reducir el coste computacional del algoritmo a costa de perder precisión. Por ejemplo, en el caso de Zernike puede reducirse el número de polinomios utilizados. En el caso de MVCNN, puede reducirse el tamaño final de las características. ▪ Reducir el impacto del efecto: Se puede conservar el uso de las características de Zernike, pero estas deberán ser calculadas en el servidor central y no podrá realizarse en paralelo desde las cámaras. Esto implica que las cámaras tendrán que volver a enviar las vistas hasta el servidor.

Tabla 6.3: Riesgo de que los tiempos sean inasequibles al calcularlos en las placas vinculadas a cada cámara.

Nombre	El sistema de clasificación funcione mejor de lo esperado en producción.
Descripción	Durante la experimentación se ha seleccionado una base de datos especialmente compleja, con muchas clases distintas y similares entre si. Es posible el cliente que incorpore Zerogravity3D tenga un conjunto de piezas más diferenciable que el utilizado durante la experimentación.

Causa	La base de datos utilizada no corresponde a ningún caso de uso particular, es decir, se ha buscado que sea genérica y amplia.
Evento	Una vez realizadas las pruebas con las piezas del cliente, se determina que la precisión obtenida es superior a la esperada.
Efecto	Mayor gratificación para el cliente y posibilidad de estar interesado en otros productos.
Evaluación la oportunidad	▪ Probabilidad: 10 %. Se ha utilizado una base de datos extensa y variada para extraer los resultados. ▪ Impacto: 70 %. Mejoraría el producto y la satisfacción del cliente. ▪ Riesgo: 0.07 (Tolerable).
Respuesta	Para explotar esta oportunidad es necesario realizar pruebas dentro de un caso de uso real. Esto es, adquirir piezas reales (a diferencia de la simulación que se ha realizado en la experimentación) y evaluar la precisión.

Tabla 6.4: Oportunidad de que el sistema funcione mejor de lo esperado.

6.3 Identificación y análisis de las soluciones propuestas

En la sección *"5. Comparativa"* se proponen dos alternativas posibles: utilizar Zernike como extractor de características o MVCNN. Ambos dan unas tasas de acierto aceptables (alrededor el 90 %) y tiempos de cómputo razonables. Sin embargo, y como se menciona en la sección 6, MVCNN es una técnica costosa de realizar, y puede ser inasequible en placas con recursos limitados.

Además, Zernike tiene una ventaja adicional, y es que su extracción de características no es necesario de ser entrenada previamente y, por tanto, se elimina el riesgo de que la red no converja (riesgo recogido en la sección 6). No solo esto, si se incluyera una pieza nueva y se utilizara MVCNN, sería necesario volver a entrenar la red con el nuevo conjunto de datos de entrenamiento. Es necesario recordar que además de la fase de entrenamiento dentro del clasificador con rechazo, MVCNN necesita ajustar los pesos de la red con un entrenamiento propio (véase sección 4.6.2) para que posteriormente extraiga correctamente las características de las vistas. Esto requeriría un sistema de cómputo dentro de la fábrica capaz de permitir el entrenamiento de la red, que suele tener unos requerimientos costosos. Este entrenamiento, además, suele requerir un tiempo de parada del sistema elevado.

Zernike, al no ser una técnica de *Deep Learning* y no tener que ajustar pesos, extrae directamente las características sin necesidad de ningún entrenamiento. Si se añade una pieza nueva a la fabricación, el único entrenamiento necesario sería incluir esa nueva clase, con sus correspondientes características, dentro del clasificador con rechazo. No es necesario modificar las características de las clases que ya están en producción. Por tanto, la opción más razonable es implantar Zernike como extractor de características.

CAPÍTULO 7

Solución propuesta

7.1 Plan de trabajo

La experimentación incluída en este trabajo se ha realizado en un entorno aislado al funcionamiento de Zerogravity3D. Por tanto, es necesario migrar el código implementado de la solución propuesta a un sistema más eficiente. Esta migración será al lenguaje C++, ya que además es el lenguaje en el cual está implementada toda la funcionalidad de Zerogravity3D.

Este proceso de migración y preparación de la solución se describe en la figura 7.1. En esta se desglosa el proceso en pequeñas tareas, que son:

- Implementación en C++.
- Puesta a punto de las placas.
  - Instalación de las placas.
  - Pruebas de conexión de las placas.
- Compilación de una base de datos. Esto se realizaría, a diferencia de en la experimentación, con piezas reales que serían capturadas mediante Zerogravity3D.
- Entrenamiento del modelo.
- Pruebas de tiempo.
- Pruebas de precisión con la nueva base de datos.

Se estima una participación de dos personas para toda la implantación. En la figura 7.1 se separa mediante colores la contribución esperada de cada persona. Se

requiere de una persona (será referido como "Trabajador 1") con los conocimientos necesarios en C++ para la migración y que esté familiarizada con el sistema de clasificación. Por otra parte, se requiere de una segunda persona ("Trabajador 2") que se encargue de la elaboración del sistema de las placas, su comunicación con el servidor central y las pruebas de conexión. El tiempo estimado del proceso es 1 mes y 10 días, contando días festivos.

7.2 Presupuesto

Por fortuna, el prototipo de Zerogravity3D ya cuenta con toda la infraestructura sobre la que instalar las cámaras y únicamente resta adquirir las nuevas placas vinculadas a cada cámara y sustituirlas por las actuales. Por esto, se debe destinar parte del presupuesto a adquirir las nuevas cámaras NVIDIA JETSON NANO. Se han seleccionado estas gracias a un estudio de requisitos previo realizado dentro del Instituto Tecnológico de Informática.

Cada NVIDIA JETSON NANO tiene un valor de 109 €.[1] Zerogravity3D tiene 16 cámaras, por lo que el coste de adquirir todas las placas de las cámaras asciende a 1744 €.

Por otra parte, se requiere especificar la inversión que debe ser realizada en recursos humanos. Tal y como se estima en 7.1 se requieren dos personas para realizar este proyecto y tendrá una duración de 1 mes y 10 días. Según la resolución de 16 de octubre de 2019, de la Dirección General de Trabajo, por la que se registran y publican las tablas salariales para 2019 del XIII Convenio colectivo de ámbito estatal para los centros de educación universitaria e investigación, el salario de un Técnico de TIC es de 1246 € mensuales. Esto da lugar a un salario anual de 17445 €. El salario en el Instituto Tecnológico de Informática está prorrateado a 12 meses. Por tanto, el salario mensual de cada trabajador sería 1454 €.

El primer trabajador, marcado en color verde en la figura 7.1, tiene una dedicación completa de 1 mes y 10 días. Esto resulta en un coste aproximado de 1950 €. El segundo trabajador, marcado en rojo, tiene una dedicación completa pero de menor duración (17 días). Esto implica un coste de 850 €.

[1] https://www.nvidia.com/es-es/autonomous-machines/embedded-systems/jetson-nano/

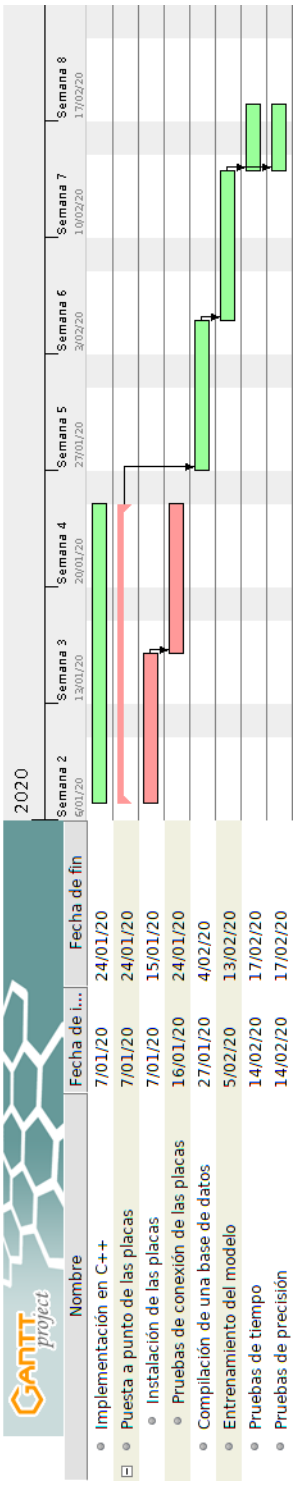


Figura 7.1: Diagrama de Gantt estimado para la implementación de la solución.

En resumen, el coste total del proyecto es de 4544 €. La tabla 7.1 recoge el conjunto de los costes del proyecto.

Recursos	Costes
Placas NVIDIA	1744 €
Trabajador 1	1950 €
Trabajador 2	850 €
Total	4544 €

Tabla 7.1: Tabla de presupuestos para el proyecto.

7.3 Diseño de la solución

La solución propuesta supone un cambio en la estructura de Zerogravity3D. Actualmente, las piezas son proyectadas dentro de Zerogravity3D y capturadas cuando llegan al centro de este. Cada cámara extrae una de las vistas del objeto y la envían hasta un servidor central. En este se realizan todas las operaciones posteriores necesarias para la clasificación. Primero se reconstruye el objeto y se genera un modelo 3D. De este modelo 3D se extraen las características 3D que serán utilizadas para la clasificación. En la figura 7.2 se muestra este flujo de los datos.

Sin embargo, gracias a la introducción del uso de características de las vistas, este proceso es más reducido. De la misma forma que en la implantación anterior, se obtienen las vistas del objeto proyectado. Sin embargo, las vistas del objeto ya no se envían al servidor central, si no que se aplica el preprocesado (descrito en la sección 4.5) para, posteriormente, realizar la extracción de características en la propia placa vinculada a la cámara. De esta forma, al servidor central únicamente le llegan las características extraídas de las vistas. Este servidor central almacena los distintos clasificadores ya entrenados, es decir, el clasificador por tamaño y el clasificador con rechazo. De esta forma, el servidor central se encarga únicamente del cómputo final de la clase de la pieza. Además, esto implica que existe un cierto nivel de paralelización; las placas vinculadas a las cámaras pueden calcular las características de la pieza siguiente mientras el servidor central realiza la clasificación de la pieza actual. La figura 7.3 muestra el nuevo flujo de datos de Zerogravity3D propuesto.

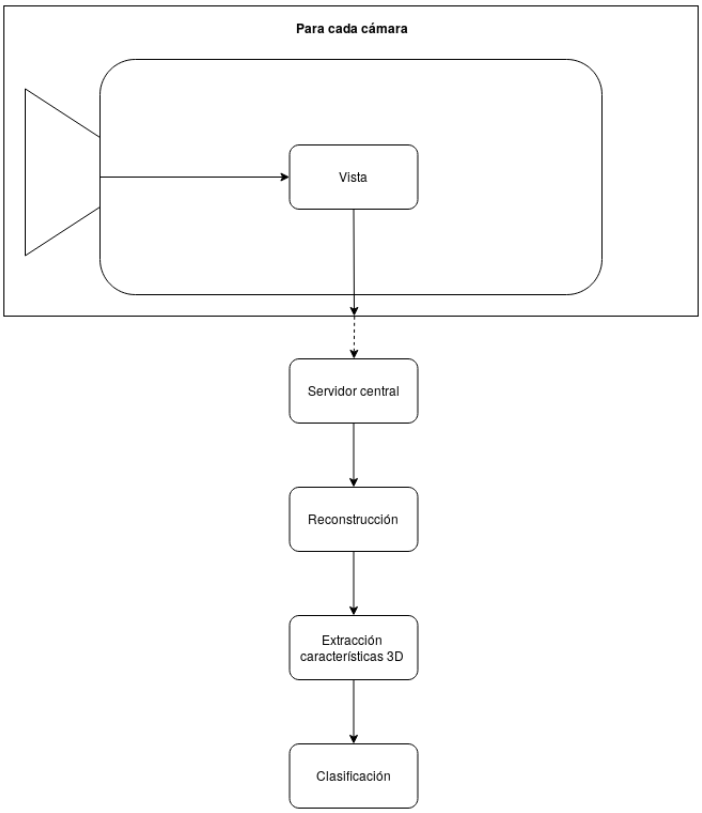


Figura 7.2: Estructura actual de Zerogravity3D.

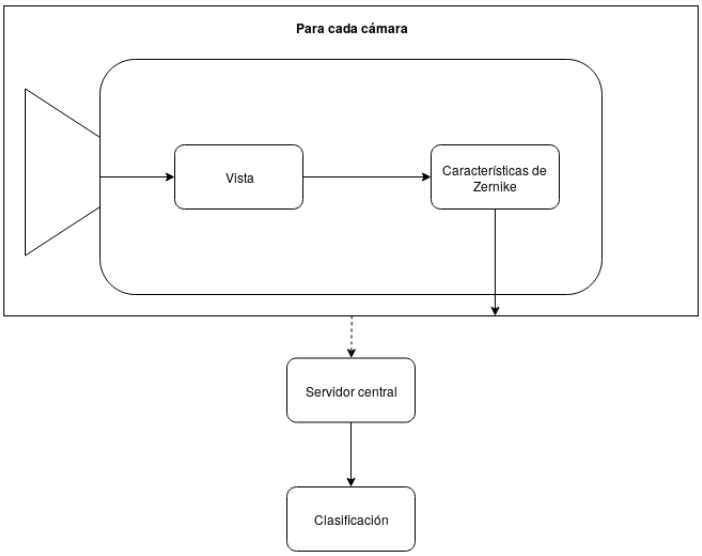


Figura 7.3: Estructura propuesta.

7.4 Tecnología utilizada

A lo largo de todo el trabajo el código ha sido desarrollado en Python. En la implementación de Zernike se han utilizado varias librerías para ayudar con el cómputo de operaciones matriciales como Numpy, y Scikit-learn para ayudar con varios cálculos matemáticos, como el factorial de un conjunto de valores. Además, se ha utilizado la versión de *KDTree* de Scipy para la eficiente búsqueda de vecinos más cercanos en el clasificador con rechazo. Es crucial que este algoritmo sea lo más eficiente posible ya que, cuando el número de piezas del conjunto de entrenamiento aumenta, también aumenta el tiempo de cómputo de los vecinos más cercanos. También se han realizado paralelizaciones del código mediante librerías como multiprocessing. Sin embargo, en la implantación final se busca la máxima eficiencia, y Python tiende a no cumplir este requisito debido a su naturaleza de lenguaje interpretado.

Para la implementación final es necesario migrar el código a C++ para tener una mayor eficiencia en el cómputo de las características. Además, el código de Zerogravity3D se encuentra en C++, por lo que esta migración mejora la mantenibilidad del producto. En la implementación de C++ se espera utilizar librerías como OpenCV para el manejo de las imágenes, y a su vez librerías para la optimización del código como OMP o CUDA. Ya que las placas vinculadas a las cámaras son NVIDIA JETSON NANO, estas cuentan con un procesador gráfico y, por tanto, el algoritmo de extracción de características de Zernike será paralelizado en los múltiples núcleos de la gráfica. Se puede utilizar OMP para generar hilos que optimicen los tiempos de cómputo de tareas costosas, independientes entre sí, que pueden surgir en el clasificador, como por ejemplo la búsqueda en paralelo de los vecinos más cercanos de varios conjuntos de características en el clasificador con rechazo. Se recomienda el uso de sistemas operativos libres como cualquier distribución de Linux por varios motivos:

- Cada placa vinculada a una cámara debe de tener un sistema operativo, y es costoso obtener una licencia para cada dispositivo.
- El sistema operativo instalado debe de estar preparado para sistemas de cómputo con recursos reducidos. Esto es, este debe de ser ligero en el uso de memoria y minimizar su uso de los recursos de cómputo.

Actualmente, las placas tienen instalado el sistema operativo por defecto; este es, Linux 16.04.

CAPÍTULO 8

Conclusiones

Tras una comparativa entre distintos extractores de características, MVCNN y Zernike han resultado ser los dos mejores para este conjunto de datos. Por otra parte, tanto RLBP como su combinación con Zernike no consiguen estar a la altura de los otros extractores de características, aunque sí que consiguen el menor tiempo de cómputo.

En cuanto a la extracción de características del modelo 3D, se ha demostrado que añadir el histograma de distancias al centroide mejora los resultados frente al clasificador original. Aun así, comparado con los extractores de características de las imágenes del modelo, estas no consiguen una tasa de acierto a la altura de MVCNN y Zernike, aunque si que consiguen una tasa de error baja.

La mejora en precisión frente a las características 3D implica que, para la clasificación con rechazo, no es necesario realizar la etapa de reconstrucción dentro del sistema de Zerogravity3D, permitiendo reducir el coste temporal aumentando la precisión.

Finalmente tras un análisis de los riesgos, se escoge Zernike frente a MVCNN como extractor de características debido a los problemas de entrenamiento que puede ocasionar MVCNN y el mayor coste computacional.

8.1 Reflexión

Debido a la extensión del trabajo, especialmente de la experimentación, no ha sido posible realizar la implementación de la solución propuesta. Esto queda, por

tanto, registrado como un objetivo posterior a la finalización de este trabajo.

Este trabajo busca aunar campos variados de la informática, sobre los cuales resalta la Inteligencia Artificial, aunque sin descuidar la parte de gestión de proyectos, y organización, optimización y testing de código. Las tres últimas tienden a ser las más invisibles pero reducen significativamente el esfuerzo necesario para realizar este trabajo.

En lo conceptual, lo más complejo ha sido aprender los conocimientos matemáticos detrás de cada extractor de características, y muchos otros que no han sido incluidos finalmente, así como aprender lo necesario del campo de visión por computador, que ocupa gran parte de la parte teórica del trabajo.

En lo temporal, lo más costoso ha sido la experimentación ya que se ha tenido que implementar la estructura de código necesaria para realizar la comparativa. Hay multitud de procesos que sufren las piezas desde el modelo 3D hasta los resultados finales que han tenido que ser implementados desde cero. Además, la red convolucional MVCNN requiere de una elevada capacidad de cómputo y tiempo, lo cual ha sido bloqueante en algunas ocasiones.

CAPÍTULO 9

Trabajo futuro

- La publicación de MVCNN [21] especifica que se realiza una concatenación de las características de las vistas, ya que es necesario para que la clasificación mediante *MLP* sea correcta. Si no se concatenaran las vistas de MVCNN, el orden en el que se posicionan las vistas sería determinante y los resultados obtenidos serían distintos. A diferencia del trabajo de MVCNN, este no utiliza *MLP*, por lo que las vistas podrían ser utilizadas por separado como se realiza con Zernike y RLBP. Esto podría dar mejores resultados y una mayor separabilidad para las piezas que son demasiado distintas de la clase original.
- Optimizar la solución propuesta. Los tiempos han sido medidos utilizando Python. Si se realizara una migración a C++ se podrían obtener tiempos inferiores.
- Implementar la solución propuesta en Zerogravity3D.
- Estudiar las múltiples técnicas para garantizar que el entrenamiento de la red MVCNN extrae las características óptimas para el conjunto de piezas. De esta forma se podría implantar MVCNN ya que es más preciso que Zernike.
- Eliminar el clasificador por tamaños e incluir ese valor como una característica más.

CAPÍTULO 10

Relación con los estudios cursados

Debido a que este trabajo ha sido realizado como trabajo final del Máster Universitario en Ingeniería Informática, es importante realizar una introspección y resaltar los conocimientos adquiridos en este que han ayudado en gran medida a la elaboración de este proyecto. Se buscan tanto los conocimientos técnicos acerca del tema tratado, como los de gestión de proyectos, de sistemas de procesamiento y testing del código.

De la asignatura de SIN ha sido de gran uso la parte de las prácticas con respecto a técnicas de preprocesado y búsqueda de características, además de los conocimientos adicionales en optimización.

De la asignatura de COS se ha utilizado la configuración de los sistemas de cómputo para utilizar sistemas de almacenamiento distribuidos, así como todo lo relacionado con el uso de SSH para conectarse al servidor remoto y para el traspaso de datos.

De la asignatura de ACG se ha extraído la importancia de un código sólido y bien testeado con el fin de ahorrarse tiempo en el futuro. Se han implementado tests unitarios que tenían en cuenta no solo los casos comunes de uso, sino también la respuesta frente valores extremos.

De la asignatura de CAP se han obtenido los conocimientos para la gestión de múltiples procesos e hilos para reducir los elevados tiempos de cómputo que tenía este trabajo, así como la gestión de CUDA.

De la asignatura de PdP se han extraído los conocimientos para la elaboración de un análisis y gestión de los riesgos vinculados al proyecto y de la planificación de este.

10.1 Soft skills

De entre todos los soft skills que aporta el máster MUIINF, cabe destacar dos que han sido los más presentes en este trabajo: el análisis y resolución de problemas, y la planificación y gestión del tiempo. El primero ha sido más significativo a la hora de buscar entre todo el estado del arte las técnicas que mejor se ajusten al problema dadas sus restricciones. El segundo ha sido notable a lo largo de todo el proyecto, gestionando mi dedicación al proyecto y a otras tareas que debía realizar en el Instituto Tecnológico de Informática, para cumplir los plazos establecidos en ambos. Esto me permitía avanzar en paralelo múltiples proyectos, estimar tiempos y planificar mejor.

CAPÍTULO 11

Agradecimientos

Agradecer a Javier Cano, jefe de proyectos del departamento de PRAIA dentro del Instituto Tecnológico de la Informática, por permitirme realizar este trabajo.

Gracias Juan Carlos Pérez, el tutor de este trabajo, y a Jose Luis Guardiola por haberme ayudado tanto con sus enormes conocimientos y experiencia dentro de este campo.

Gracias a Paula Sánchez Cuevas por recibir siempre su inmenso apoyo, su paciencia y por ayudarme a mejorar como redactor y como persona.

CAPÍTULO 12

Glosario

Data Augmentation
Técnica usada en Machine Learning para aumentar el número de muestras de la base de datos realizando modificaciones coherentes que añadan la suficiente variación como para ser consideradas muestras independientes.

Backpropagation o propagación hacia atrás
Método utilizado para el cálculo del gradiente utilizado principalmente en el entrenamiento de las redes neuronales. Este propaga el gradiente desde la última capa hasta la primera. Gracias a esto, los pesos se ajustan iterativamente para minimizar el error.

Multi Layer Perceptron (MLP)
Red neuronal tradicional de múltiples capas. Iterativamente, hasta alcanzar la última capa, multiplica la entrada de la capa por un vector de pesos y aplica una función no lineal al resultado. Estos pesos transforman el vector de entrada a un vector de salida. Estos se ajustan durante la fase de entrenamiento para obtener el resultado esperado en el vector de salida. El ajuste se realiza mediante *backpropagation*.

Regresión
Dentro del Machine Learning se pueden resaltar, de entre otros, dos tipos de problemas: los problemas de regresión y los problemas de clasificación. Los problemas de regresión son aquellos en los que la salida del algoritmo se espera un valor continuo (como por ejemplo en la predicción del consumo de energía del siguiente día, se espera un valor real continuo), frente a la tarea de clasificación que espera salidas con valores discretos (como, por ejemplo, qué clase de pieza

es la que aparece en esta imagen).

Learning Rate
Durante el *backpropagation* es interesante regular con cuanta intensidad se van a modificar los pesos con el gradiente. Si este valor es muy grande, cabe la posibilidad de que el descenso por gradiente salte un mínimo óptimo, o incluso que se quede dando saltos y no converja. Para esto último se tiene que imaginar una función con forma de U en la cual el punto óptimo es la parte inferior de la U, pero el algoritmo, en vez de ir progresivamente bajando hasta llegar al mínimo, salta de un lado a otro.

Adam
Optimizador de redes neuronales que se encarga de ajustar el *learning rate* asignado al entrenamiento dinámicamente. Es capaz de prever si, dado el punto actual del entrenamiento, se necesita un *learning rate* mayor o menor.

Random Search
Método de optimización de hiperparámetros que consiste en probar valores aleatorios dentro de un rango acotado y obtener la combinación que minimice (o maximice) una función objetivo.

Weight Decay
Tipo de regularización utilizada en redes neuronales para evitar que esta memorice las muestras de entrenamiento y disminuya su capacidad de generalización (también llamado overtraining).

K-Nearest Neighbours (KNN)
Algoritmo de clasificación supervisado que está basado en la premisa de que muestras similares (porque, por ejemplo, pertenezcan a la misma clase) se hallan juntas en el espacio. Por ejemplo, en una tarea de clasificación en la cual nos interesa saber si la muestra que tenemos es de la clase A o la clase B, podríamos buscar cuales son los K prototipos de entrenamiento más cercanos a la nueva muestra y asignarles la clase mayoritaria.

Deep Learning
Campo dentro del Machine Learning que estudia cómo las redes neuronales, generalmente de gran tamaño, son capaces de aprender distintas tareas de Machine

Learning gracias a su gran capacidad. Generalmente son técnicas bastante costosas computacionalmente y suelen requerir del uso de GPUs.

Prototipo
Termino utilizado para referirse a una muestra de entrenamiento.

Producto escalar de Frobenius
En el caso de este trabajo, el producto escalar de Frobenius, representado mediante $\langle \cdot, \cdot \rangle_F$, consiste en multiplicar dos matrices elemento por elemento y realizar la suma de los resultados. Cada celda se multiplica con la celda correspondiente de la otra matriz y, finalmente, se suman todos los resultados obtenidos. Esto implica que ambas matrices deben tener las mismas dimensiones.

Outlier
Objeto que no puede pertenecer a ninguna clase que esté siendo considerada debido a la significativa diferencia entre sus características y las características esperadas. Por ejemplo, un outlier puede ser un muelle en una línea de producción en la que se estén procesando tornillos.

KDTree
Es una clase de árbol binario que se utiliza como una estructura para la búsqueda eficiente de elementos. Esta estructura divide el espacio en bloques, en concreto, cada nodo del árbol divide es espacio en dos. Esto permite determinar para cada nodo a qué parte del espacio pertenece una muestra dada mediante una rápida comparación. El uso más común dentro de este trabajo es para construir una estructura que contiene todos prototipos de entrenamiento. Dada esta estructura, se puede calcular rápidamente los vecinos más cercanos de una nueva pieza ya que esta los busca dentro del subconjunto que existe en su bloque del espacio.

Bibliografía

[1] Timo Ahonen, Jiří Matas, Chu He, and Matti Pietikäinen. Rotation invariant image description with local binary pattern histogram fourier features. In Arnt-Børre Salberg, Jon Yngve Hardeberg, and Robert Jenssen, editors, *Image Analysis*, pages 61–70, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[2] Joaquim Arlandis, Juan-Carlos Perez-Cortes, and Javier Cano-Perez. Rejection strategies and confidence measures for a k-nn classifier in an ocr task. 16, 08 2002.

[3] Paul Bergmann, Michael Fauser, David Sattlegger, and Carsten Steger. Mvtec ad – a comprehensive real-world dataset for unsupervised anomaly detection. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[4] Paul Bergmann, Sindy Löwe, Michael Fauser, David Sattlegger, and Carsten Steger. Improving unsupervised defect segmentation by applying structural similarity to autoencoders. *Proceedings of the 14th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*, 2019.

[5] Pau Garrigues Carbó. Diseño e implementación de un clasificador mediante redes neuronales par aun sistema de inspección industrial 3d. 2019.

[6] Ken Chatfield, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Return of the devil in the details: Delving deep into convolutional nets. *Proceedings of the British Machine Vision Conference 2014*, 2014.

[7] Ahmad Basheer Hassanat, Mohammad Ali Abbadi, Ghada Awad Altarawneh, and Ahmad Ali Alhasanat. Solving the problem of the k parameter in the knn classifier using an ensemble learning approach, 2014.

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2016.

[9] Rajesh Kumar Muniandy Ho Wei Yong, Abdullah Bade. 3d reconstruction of breast cancer from mammograms using volume rendering techniques. *Jurnal Teknologi*, 75(2).

[10] M. Jirina and M. Jirina. Using singularity exponent in distance based classifier. In *2010 10th International Conference on Intelligent Systems Design and Applications*, pages 220–224, Nov 2010.

[11] Marcel Jirina, MJ Jirina, and K Funatsu. Classifiers based on inverted distances. In *New Fundamental Technologies in Data Mining*, volume 1, pages 369–387. InTech, 2011.

[12] Marcel Jiřina and Marcel Jiřina jr. Classifier based on inverted indexes of neighbors. Technical report, Technical Report No, 2008.

[13] A. Khotanzad and Y. H. Hong. Invariant image recognition by zernike moments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(5):489–497, May 1990.

[14] Graciela Lara López, Adriana Peña Pérez Negrón, Angélica De Antonio Jiménez, Jaime Ramírez Rodríguez, and Ricardo Imbert Paredes. Comparative analysis of shape descriptors for 3d objects. *Multimedia Tools and Applications*, 76(5):6993–7040, Mar 2017.

[15] D. A. Lisin, M. A. Mattar, M. B. Blaschko, E. G. Learned-Miller, and M. C. Benfield. Combining local and global image features for object class recognition. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Workshops*, pages 47–47, Sep. 2005.

[16] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, Nov 2004.

[17] Andrew Z Luo, Eric Whitmire, James W Stout, Drew Martenson, and Shwetak Patel. Automatic characterization of user errors in spirometry. In *Engineering in Medicine and Biology Society (EMBC), 2017 39th Annual International Conference of the IEEE*, pages 4239–4242. IEEE, 2017.

[18] Rakesh Mehta and Karen O. Egiazarian. Rotated local binary pattern (rlbp) - rotation invariant texture descriptor. In *ICPRAM*, 2013.

[19] Timo Ojala, Matti Pietikäinen, and David Harwood. A comparative study of texture measures with classification based on featured distributions. *Pattern Recognition*, 29(1):51 – 59, 1996.

[20] Benjamin Staar, Michael Lütjen, and Michael Freitag. Anomaly detection with convolutional neural networks for industrial surface inspection. *Proce-*

dia CIRP, 79:484 – 489, 2019. 12th CIRP Conference on Intelligent Computation in Manufacturing Engineering, 18-20 July 2018, Gulf of Naples, Italy.

[21] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In *Proceedings of the IEEE international conference on computer vision*, pages 945–953, 2015.

[22] Wikipedia. 3d reconstruction from multiple images.

[23] Sheng-Chih Yang, Cheng-Yi Yu, Cheng-Jian Lin, Hsueh-Yi Lin, and Chi-Yuan Lin. Reconstruction of three-dimensional breast-tumor model using multispectral gradient vector flow snake method. *Journal of Applied Research and Technology*, 13(2):279 – 290, 2015.

[24] Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. *Lecture Notes in Computer Science*, page 818–833, 2014.