



Visión por Computador 2024-2025

MASTER CIENCIA DE DATOS

UNIVERSIDAD DE GRANADA

# La Visión por Computador: Historia, Resumen y Aplicaciones Actuales

MIGUEL GARCÍA LÓPEZ

# Índice

<b>1. Introducción</b>	<b>4</b>
<b>2. Visión humana</b>	<b>4</b>
2.1. El Sistema Visual . . . . .	4
2.2. Estructura del Ojo . . . . .	5
2.3. La Retina . . . . .	5
2.3.1. Composición Celular . . . . .	5
2.3.2. Distribución de Fotorreceptores . . . . .	5
2.4. Procesamiento Visual . . . . .	6
2.4.1. Circuito Retinal . . . . .	6
2.5. Movimientos Oculares . . . . .	7
2.6. Visión Tridimensional . . . . .	7
<b>3. Historia, Influencia e Inspiración</b>	<b>7</b>
3.1. Hitos importantes . . . . .	7
<b>4. Elementos</b>	<b>12</b>
4.1. Hardware . . . . .	12
4.1.1. Dispositivo LiDAR . . . . .	13
4.1.2. Cámaras digitales . . . . .	13
4.1.3. Dispositivos de procesamiento . . . . .	14
4.2. Software . . . . .	15
4.2.1. OpenCV . . . . .	16
4.3. PyTorch . . . . .	16
4.4. Integración . . . . .	18

<b>5. Aplicaciones actuales</b>	<b>18</b>
<b>6. Aplicación real: Detección de anomalías con <i>Deep Learning</i></b>	<b>20</b>
<b>7. Adquisición de imágenes</b>	<b>20</b>
<b>8. Preprocesamiento</b>	<b>21</b>
<b>9. Segmentación</b>	<b>22</b>
<b>10. Extracción de características</b>	<b>22</b>
<b>11. Clasificación</b>	<b>23</b>
<b>12. Bibliografía</b>	<b>25</b>

## Índice de figuras

1. El nervio óptico transmite las señales generadas por la retina al procesador de visión del cerebro. . . . .	4
2. Distribución de conos y bastones en la retina. . . . .	6
3. Experimento de <i>Hubel</i> y <i>Wiesel</i> con el gato. . . . .	8
4. Proceso de extracción de características de 2D a 3D representado y explicado por <i>Larry Roberts</i> . . . . .	9
5. Estados de la representación visual según <i>David Marr</i> . . . . .	10
6. Descriptor <i>SIFT</i> en objetos de una mesa. . . . .	11
7. Arquitectura de la red <i>AlexNet</i> . . . . .	11
8. Salto de calidad en el desafío de <i>ImageNet</i> gracias a la red <i>AlexNet</i> . . .	12
9. Sistema LiDAR de un transporte aéreo. . . . .	13
10. Ilustración seccionada de una cámara con chip. . . . .	14

11.	Arquitectura básica de una <i>GPU</i> . . . . .	15
12.	Ejemplo de integración entre componentes <i>software</i> y componentes <i>hardware</i> para un sistema de visión por computador. . . . .	19
13.	Dos objetos y una textura del conjunto de datos <i>MVTec AD</i> donde se muestran imágenes anómalas y normales. . . . .	21
14.	Segmentación de anomalías en imágenes. La primera imagen es segmentada por <i>AnoGAN</i> y la segunda por <i>Autoencoders</i> tradicionales . . . . .	22



## 1. Introducción

En el presente proyecto de la asignatura de **Visión por Computador**, el alumno explicará la historia de la visión por computador y describirá cuáles fueron los puntos de inflexión e hitos que marcaron grandes avances o conceptos clave en el desarrollo de esta rama de la inteligencia artificial.

También se narrarán los elementos necesarios para aplicarla en proyectos reales y se contarán algunas soluciones que se han ido proponiendo a lo largo del tiempo. Finalmente, se describirá el **SOTA** (*state of the art*) que resuelve un problema específico.

Con este documento se proyectará en pocas páginas al lector un resumen de lo que es la visión por computador, cómo comenzó, en qué se inspiró, qué es actualmente y qué potencial tiene.

## 2. Visión humana

La visión humana es un proceso complejo que involucra la captura de luz, su conversión en señales neurales y el procesamiento cerebral para construir una representación del entorno. A continuación, se describen sus componentes y mecanismos clave.

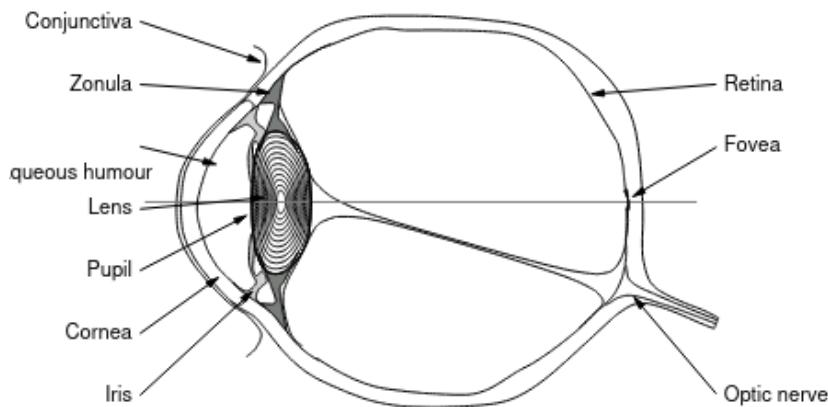


Figura 1: El nervio óptico transmite las señales generadas por la retina al procesador de visión del cerebro.

### 2.1. El Sistema Visual

El sistema visual humano puede considerarse compuesto por dos partes fundamentales: los ojos, que actúan como receptores de imágenes capturando la luz y convirtiéndola en

señales, y los centros de procesamiento visual en el cerebro, que procesan estas señales para construir una imagen interna de la escena observada.

## 2.2. Estructura del Ojo

El ojo humano presenta una estructura análoga a la de una cámara fotográfica. Sus componentes principales incluyen:

- **Córnea y humor acuoso:** Actúan como lente primaria realizando un enfoque inicial de la señal luminosa entrante.
- **Cristalino:** Controlado por el músculo denominado zónula, que ajusta tanto su forma como su posición (hacia adelante y hacia atrás), proporcionando un control fino sobre el enfoque de la luz.
- **Iris:** Músculo que, al contraerse, cubre toda la lente excepto una pequeña porción central (pupila), permitiendo un control dinámico de la cantidad de luz que ingresa al ojo.
- **Retina:** Pantalla fotosensible ubicada en la parte posterior (fig 1) del ojo, donde se enfoca la luz entrante y se convierte en señales nerviosas.
- **Fóvea:** Región central de la retina particularmente sensible debido a su alta densidad de células fotosensibles, proporcionando una excelente resolución visual.

## 2.3. La Retina

### 2.3.1. Composición Celular

La retina está compuesta por una delgada capa de células que recubre la parte posterior e interior del ojo. Contiene dos tipos principales de células fotosensibles (fig 2):

- **Bastones (Rods):** Aproximadamente 120 millones de células, muy sensibles y especializadas en la visión con poca luz.
- **Conos (Cones):** Alrededor de 6 millones de células, operan mejor en niveles normales de luz y proporcionan la visión en color.

### 2.3.2. Distribución de Fotorreceptores

La distribución de bastones y conos no es uniforme en la retina:

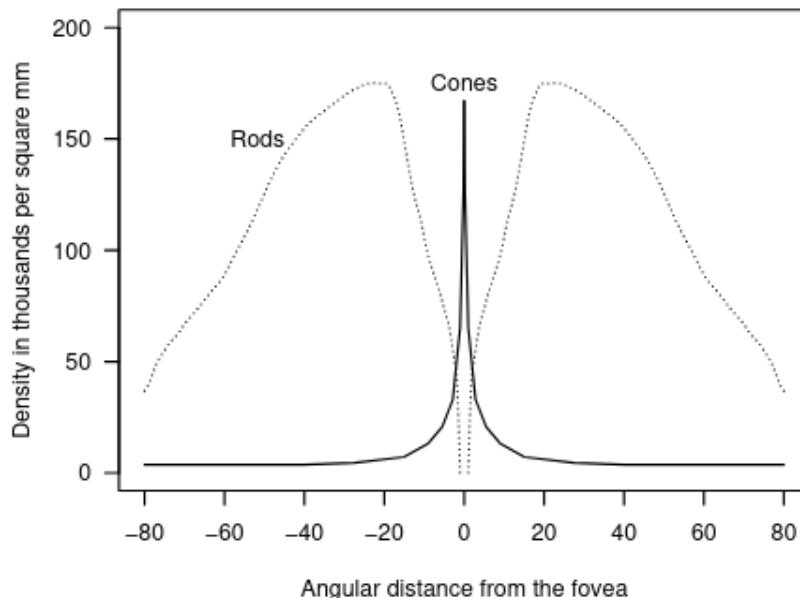


Figura 2: Distribución de conos y bastones en la retina.

- Los conos se concentran hacia el centro de la retina, especialmente en la fóvea.
- Los bastones predominan en la periferia de la retina.
- La fóvea centralis, una región de aproximadamente 0.3mm de diámetro, contiene exclusivamente conos densamente empaquetados.

## 2.4. Procesamiento Visual

### 2.4.1. Circuito Retinal

La retina realiza un procesamiento inicial de la información visual a través de diferentes tipos de células:

- **Células ganglionares:** Terminan en fibras nerviosas que conectan con el cerebro.
- **Células bipolares:** Transmiten señales de los fotorreceptores a las células ganglionares.
- **Células horizontales:** Conectan fotorreceptores adyacentes.
- **Células amacrinas:** Vinculan múltiples células ganglionares.

## 2.5. Movimientos Oculares

Los ojos realizan movimientos específicos para construir una imagen de alta resolución:

- Se mueven en series de movimientos bruscos llamados sacadas.
- Las sacadas se alternan con fijaciones estacionarias.
- Tanto las sacadas como las fijaciones duran centésimas de segundo.
- Los patrones de movimiento ocular varían según la tarea visual específica.

## 2.6. Visión Tridimensional

La percepción de profundidad se logra mediante varios mecanismos:

- **Perspectiva:** La relación entre el tamaño aparente de los objetos y su distancia.
- **Visión estereoscópica:** El procesamiento de las ligeras diferencias entre las imágenes captadas por cada ojo.
- **Oclusión:** La capacidad de objetos cercanos para ocultar objetos más distantes.
- **Efectos de iluminación:** La interpretación de sombras y gradientes de luz.
- **Efectos atmosféricos:** La interpretación de la pérdida de nitidez con la distancia.

## 3. Historia, Influencia e Inspiración

Inicialmente, los investigadores se centraron en problemas de detección de bordes, reconocimiento de patrones, etc. Fueron altamente inspirados por los sistemas de visión biológicos y trataron de desarrollar algoritmos que los imitasen [1].

### 3.1. Hitos importantes

En el año 59, *Hubel y Wiesel* [2] realizaron un experimento en el que trabajaron con un gato, al que le colocaron un sensor para detectar señales visuales (fig 3). Este sensor se ubicó en el córtex visual, y el gato fue expuesto a distintos tipos de estímulos visuales, respondiendo el sensor de diversas formas según lo que el gato veía. De esta manera, se pudieron atribuir ciertos patrones detectados a células específicas. Se considera uno de los primeros experimentos orientados a comprender cómo el cerebro procesa la información

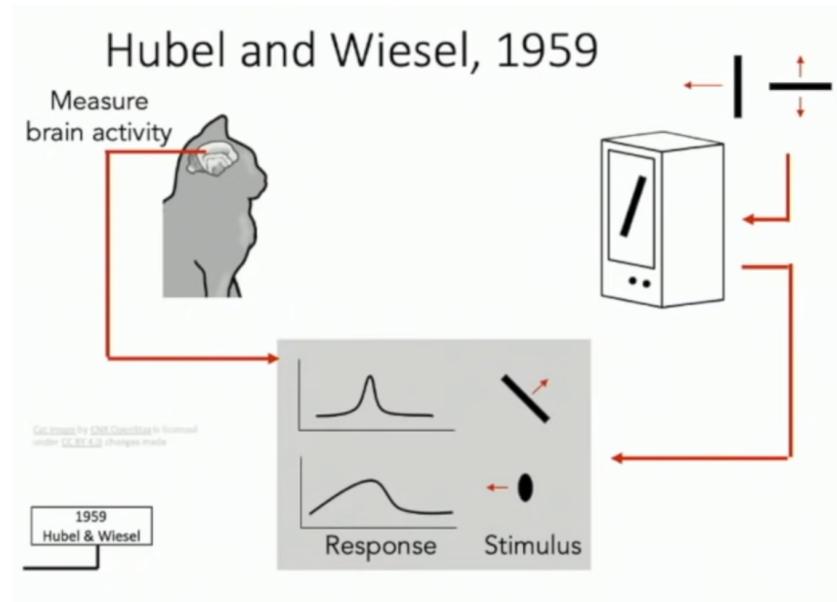


Figura 3: Experimento de *Hubel* y *Wiesel* con el gato.

visual. Se observó que células muy básicas respondían a patrones simples, mientras que combinaciones más complejas eran capaces de detectar patrones más sofisticados. De manera análoga, este principio es utilizado en las arquitecturas modernas basadas en *deep learning*.

En el año 63, *Larry Roberts* [3] presentó la primera tesis doctoral en visión por computador, enfocándose en la detección de bordes y la geometría tridimensional (3D). Su trabajo es considerado uno de los pilares fundacionales de la visión por computador. En esta tesis, *Roberts* exploró cómo una computadora podría interpretar objetos tridimensionales a partir de imágenes bidimensionales (fig 4), un problema clave en el campo. Además, propuso técnicas para extraer bordes en imágenes como un primer paso para interpretar formas tridimensionales.

En los años 70, *David Marr* [4] propuso la idea de los estados de la representación visual. En su trabajo describía cómo se dividía el proceso en:

- **Percepción de intensidades.**
- **Boceto básico:** cruces por cero, manchas, bordes, extremos, curvas, etc.
- **Boceto 2D:** orientación local de la superficie y discontinuidades en profundidad y orientación de la superficie.
- **Representación 3D:** modelos tridimensionales representados jerárquicamente en términos de **primitivas de superficie y volumétricas**. Las primitivas de superficie incluyen representaciones como triángulos, mientras que las primitivas volumétricas contienen información de volumen, como un cilindro.

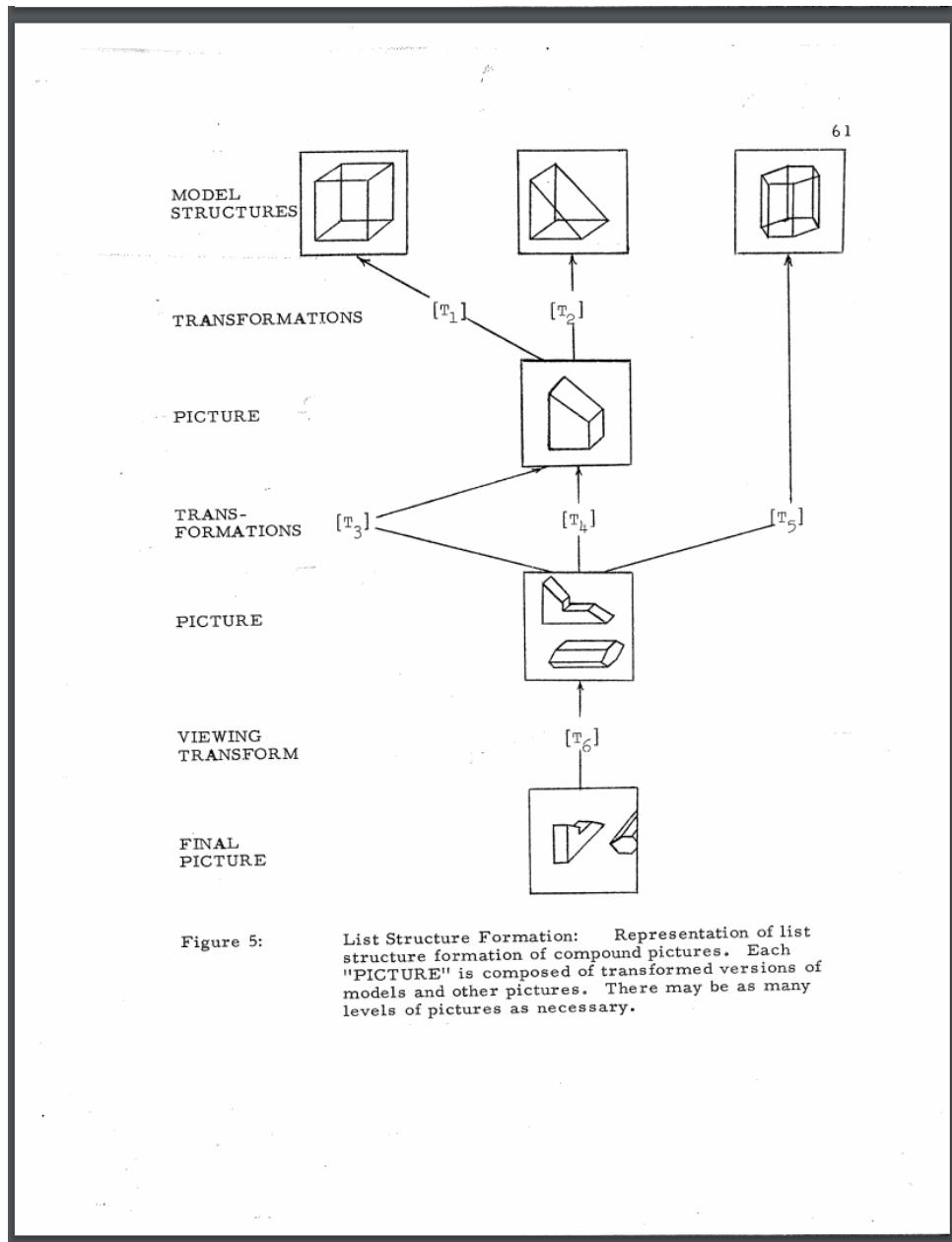


Figura 4: Proceso de extracción de características de 2D a 3D representado y explicado por *Larry Roberts*.

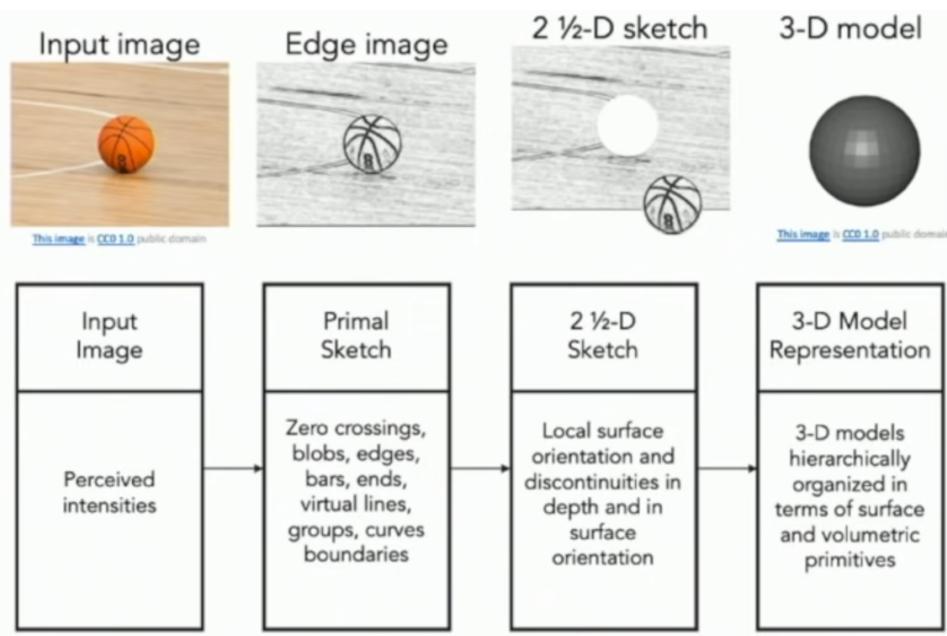


Figura 5: Estados de la representación visual según *David Marr*.

En los años 80, ya se contaba con dispositivos visuales de mayor calidad. Se comenzó a trabajar en el reconocimiento de objetos mediante la detección de bordes. Uno de los ejemplos más conocidos es el trabajo de *John Canny* [5].

En los años 90, se trabajó en reconocimiento por grupos. En los 2000, se desarrollaron los descriptores *SIFT* [6] de forma que se podían reconocer patrones mediante *matching* con estos descriptores.

- **SIFT (Transformada de Características Invariantes a la Escala)** es un algoritmo desarrollado para detectar y describir puntos clave en imágenes (fig 6).
- Es un descriptor robusto a cambios de iluminación y perspectiva.

En el año 2001, se introdujo el primer modelo basado en *machine learning* capaz de reconocer caras [7]. Este modelo fue construido utilizando el algoritmo de *Boosted Decision Trees* y se comercializó rápidamente en cámaras digitales y otros dispositivos.

El **PASCAL Visual Object Classes Challenge (PASCAL VOC)** fue una de las competiciones más influyentes en el campo de la visión por computador. Proporcionó un conjunto de datos y una serie de desafíos para evaluar algoritmos de detección y segmentación de objetos en imágenes. Contiene imágenes recolectadas de la web con 20 categorías de objetos en escenas realistas, e incluye etiquetas anotadas manualmente para cada imagen.

El **ImageNet Large Scale Visual Recognition Challenge (ILSVRC)** fue una de las competiciones más importantes en la historia de la visión por computador, ayudando

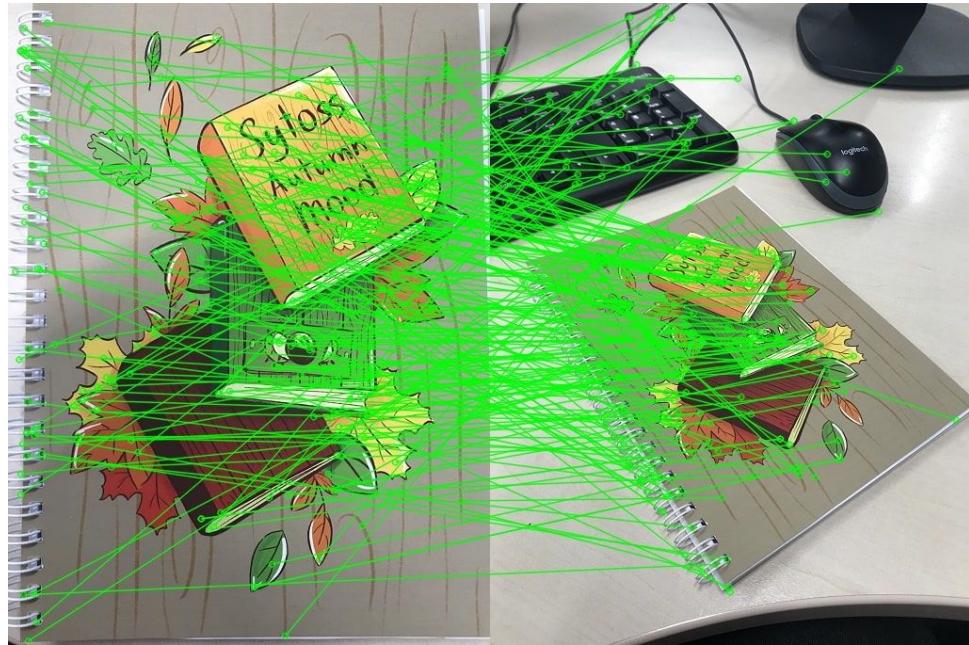


Figura 6: Descriptor *SIFT* en objetos de una mesa.

a desarrollar modelos avanzados de *deep learning*. El *dataset* de *ImageNet* incluye más de 14 millones de imágenes organizadas en alrededor de 22000 categorías basadas en la jerarquía de *WordNet*. Para la competencia *ILSVRC* se utilizó un subconjunto con 1000 clases y aproximadamente 1,2 millones de imágenes para entrenamiento, 50000 para validación y 100000 para prueba.

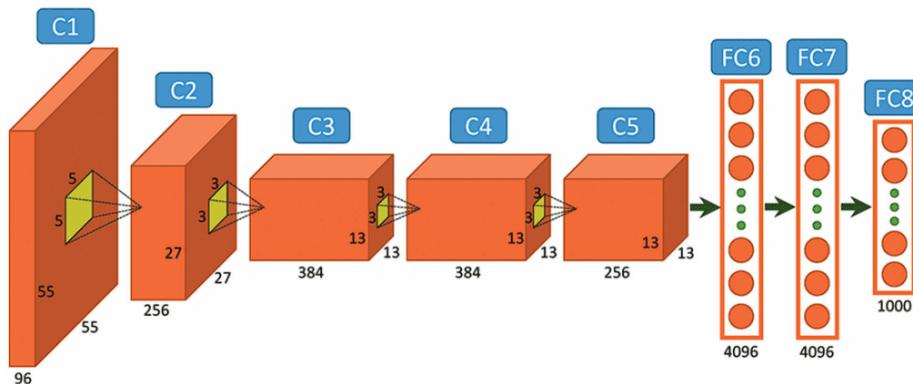


Figura 7: Arquitectura de la red *AlexNet*.

Y en 2012, apareció la red neuronal *AlexNet* [8], la cual batió récords en el desafío de reconocimiento de imágenes. Fue la primera solución de *deep learning* aplicada a este tipo de tareas. A partir de ese punto, redes neuronales más potentes fueron reduciendo el error (fig 8) del desafío de clasificación hasta llegar a superar el error medio de un ser humano. *AlexNet* (fig 7) fue una solución basada en convoluciones.

A partir de este momento en la historia, las soluciones presentadas fueron redes neu-

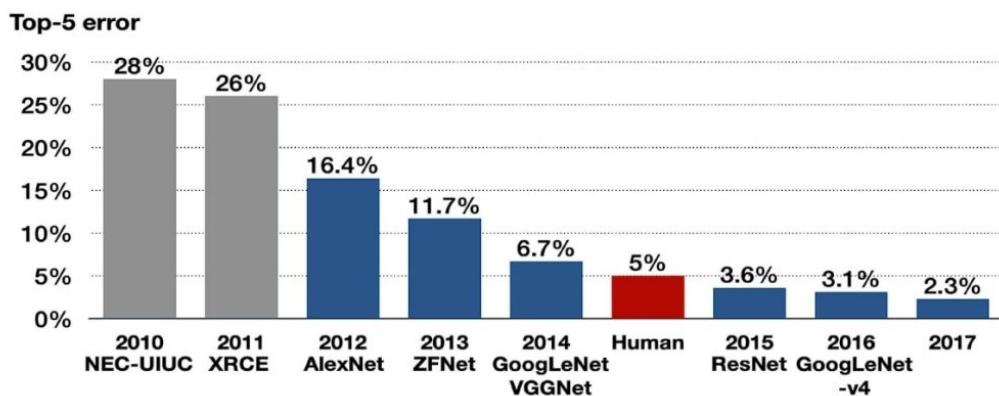


Figura 8: Salto de calidad en el desafío de *ImageNet* gracias a la red *AlexNet*.

ronales cada vez más potentes. Actualmente, las soluciones **SOTA** de visión son todas basadas en redes neuronales. El *deep learning* dominó y sigue dominando el área como la solución *de facto*.

## 4. Elementos

Un sistema de visión por computador se compone de múltiples elementos interconectados que abarcan componentes físicos, lógicos y procesos de integración. En este documento se detallará cada uno de estos aspectos.

### 4.1. Hardware

En el ámbito del *hardware*, el componente principal es el dispositivo de captura de imágenes, como cámaras digitales, sensores infrarrojos, dispositivos *LiDAR* o sistemas de radar, los cuales adquieren datos visuales o de profundidad del entorno. Estos dispositivos suelen complementarse con fuentes de iluminación controlada, filtros ópticos o sistemas de calibración para garantizar la calidad y consistencia de las imágenes.

No solo hacen falta dispositivos de captura de información visual, también de procesamiento. La información capturada se transmite a unidades de procesamiento, que pueden incluir microcontroladores, *CPUs*, *GPUs* o *TPUs*, diseñadas para manejar operaciones intensivas en cálculo, especialmente en aplicaciones en tiempo real. Además, sistemas embebidos, servidores o plataformas en la nube actúan como nodos de procesamiento remoto, dependiendo de los requisitos de latencia y capacidad. La infraestructura de almacenamiento, como memorias temporales, discos duros o bases de datos distribuidas, permite gestionar el volumen de datos generado, mientras que actuadores, robots o interfaces físicas pueden formar parte del sistema para ejecutar acciones basadas en los

resultados del análisis.

A continuación se detallan y explican algunos de estos componentes mencionados.

#### 4.1.1. Dispositivo LiDAR

Un ladar o lidar (acrónimo del inglés *LiDAR*, *Light Detection and Ranging* o *Laser Imaging Detection and Ranging*) es un dispositivo que permite determinar la distancia desde un emisor láser a un objeto o superficie utilizando un haz láser pulsado.

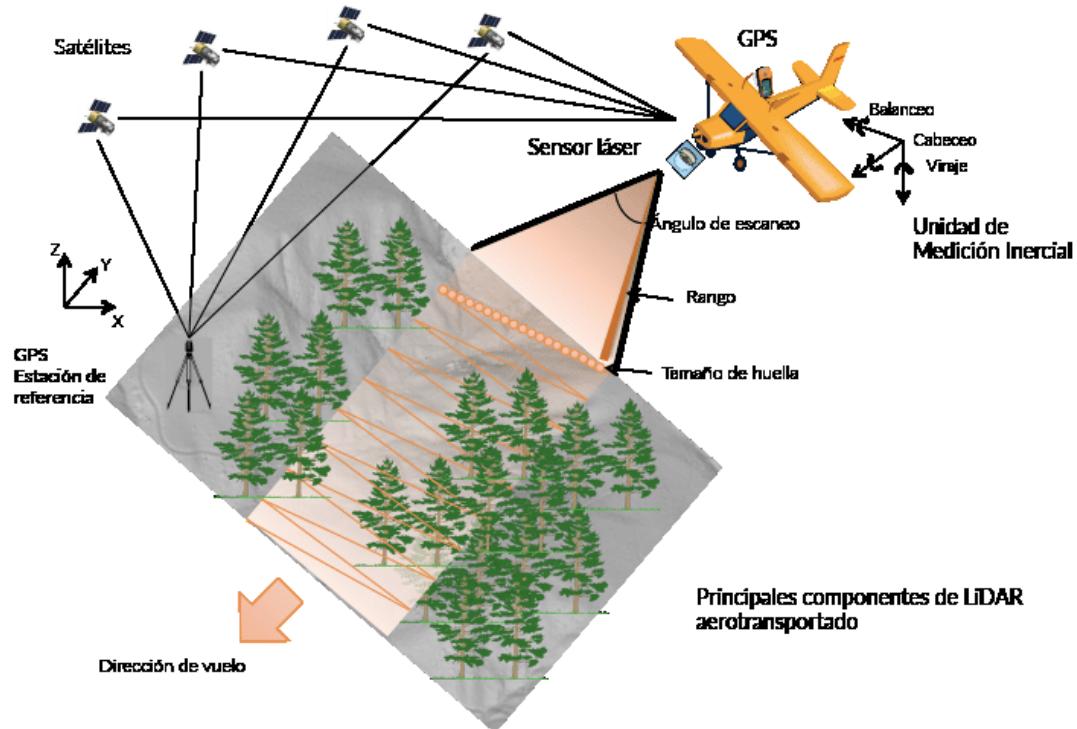


Figura 9: Sistema LiDAR de un transporte aéreo.

El ladar es un sistema que permite obtener una nube de puntos del terreno tomándolos mediante un escáner láser aerotransportado (9). Tiene múltiples usos y aplicaciones, entre ellas la topografía, mecánica de rocas, etc.

#### 4.1.2. Cámaras digitales

Possiblemente sea el tipo de dispositivo más usado en **CV**, por su coste, facilidad de uso e integración con otros dispositivos. Hoy en día, cualquier persona dispone de un dispositivo de este estilo, incluído en los teléfonos móviles y en forma de cámara fotográfica individual para los más rigurosos y amantes de la fotografía.

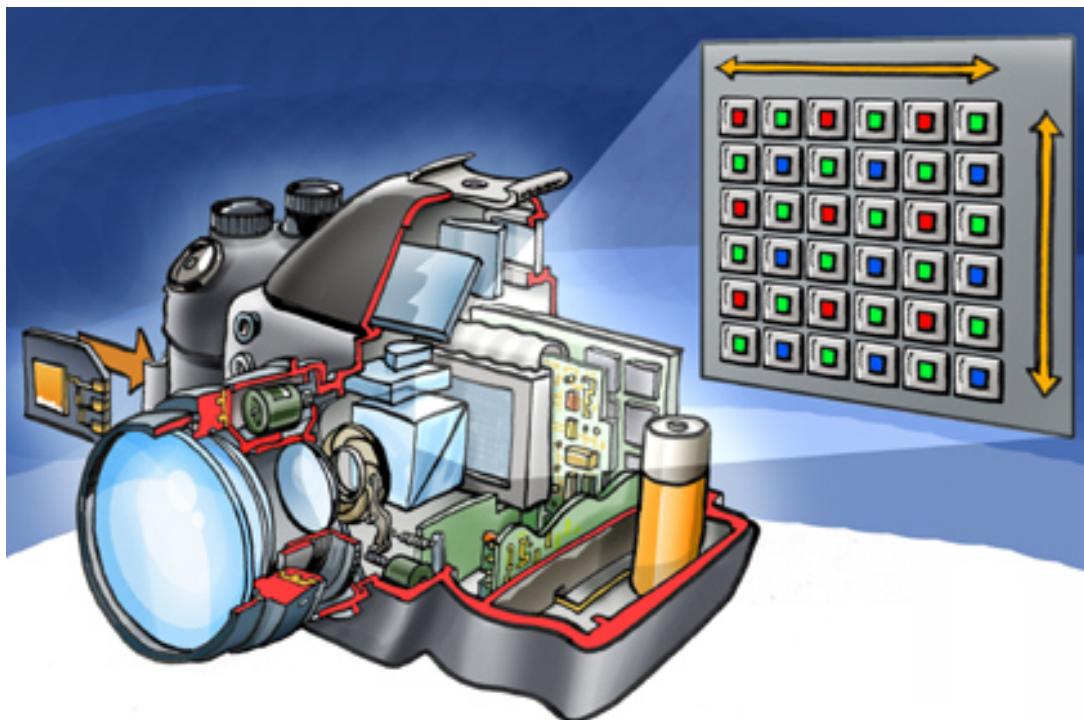


Figura 10: Ilustración seccionada de una cámara con chip.

La resolución de una cámara fotográfica digital está limitada por el sensor de la cámara (generalmente un *CCD* o un Sensor *CMOS*) que responde a las señales de luz, sustituyendo el trabajo de la película en fotografía tradicional. El sensor se compone de millones de “cubos” que se cargan en respuesta a la luz. Generalmente, estos cubos responden solamente a una gama limitada de longitudes de onda ligeras, debido a un filtro del color sobre cada uno. Cada uno de estos cubos se llama un píxel (10), y se utiliza un algoritmo de mosaicismo e interpolación para unir la imagen de cada gama de longitud de onda por píxel en una imagen del RGB donde están las tres imágenes por píxel para representar un color completo.

#### 4.1.3. Dispositivos de procesamiento

Si bien hay sistemas que utilizan *CPU*s, que son muchos, lo más utilizado a día de hoy son las unidades de procesamiento gráfico. Se pueden utilizar también las *TPU*s de *Google*, pero su acceso está restringido a servicios *online* como *Google Colab*.

Las unidades de procesamiento gráfico (fig 11) son circuitos electrónicos especializados y diseñados inicialmente para el procesamiento digital de imágenes y la aceleración de gráficos por ordenador. Tras su diseño inicial, se descubrió que las *GPU* eran útiles para cálculos no gráficos que implicaban problemas embarazosamente paralelos gracias a su estructura paralela. La capacidad de las *GPU* para realizar un gran número de cálculos con rapidez ha llevado a su adopción en diversos campos, como la inteligencia

artificial, donde destacan en el manejo de tareas de alta carga de datos y alta exigencia computacional.

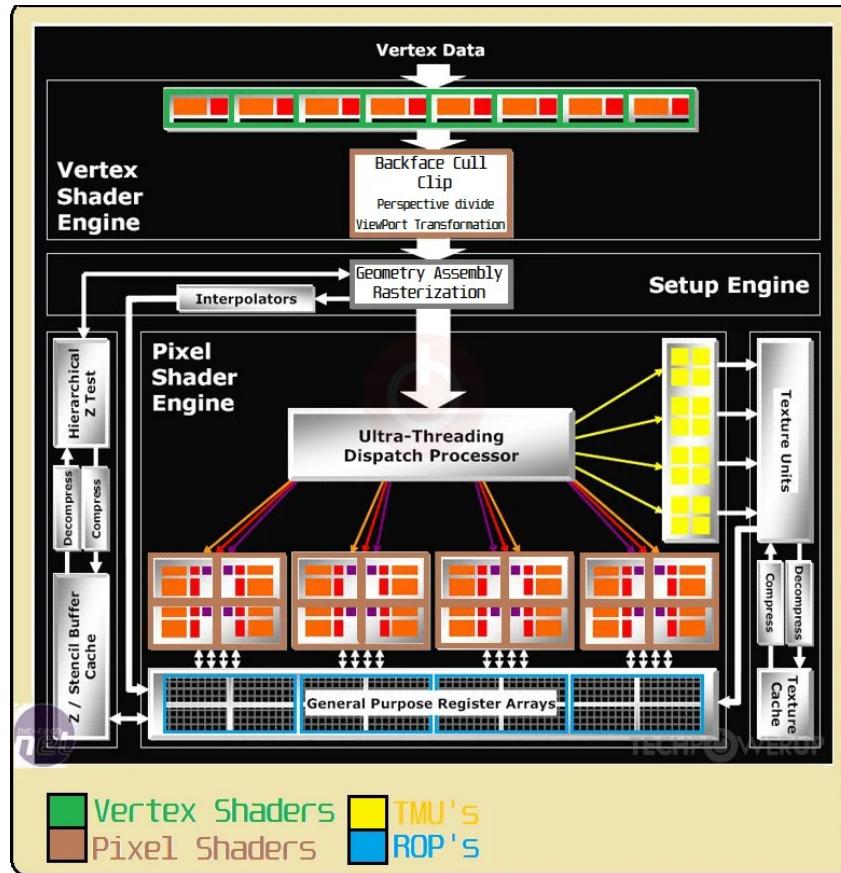


Figura 11: Arquitectura básica de una GPU.

Es por ello, que es una unidad de procesamiento excelente para el área de visión por computador, donde se trabaja con imágenes y donde las soluciones actuales involucran sistemas basados en redes neuronales.

## 4.2. Software

En la capa de *software*, el sistema se apoya en sistemas operativos y controladores específicos que gestionan la comunicación entre el *hardware* y las aplicaciones. Bibliotecas y *frameworks* especializados, como *OpenCV*, *TensorFlow*, *PyTorch* o *ROS*, proporcionan herramientas para el procesamiento de imágenes, la implementación de algoritmos de aprendizaje automático y la integración con dispositivos periféricos.

Los algoritmos abarcan desde técnicas clásicas de procesamiento de señales, como filtrado, segmentación o detección de bordes, hasta modelos avanzados de redes neuronales convolucionales (*CNN*) o *transformers*, entrenados para tareas de clasificación, detección de objetos o reconstrucción 3D.

El *software* también incluye módulos de preprocesamiento para normalizar datos, corregir distorsiones o aumentar conjuntos de imágenes, así como *pipelines* de postprocesamiento que interpretan salidas de modelos, aplican umbrales o generan visualizaciones. Interfaces de usuario, *APIs* o servicios *web* facilitan la interacción con operadores humanos o sistemas externos, permitiendo la configuración, monitoreo y despliegue de soluciones.

A continuación se detallan dos librerías clave en el desarrollo de soluciones punteras de redes neuronales y para el procesamiento de imágenes. Estas son *OpenCV* y *PyTorch*.

#### 4.2.1. OpenCV

*OpenCV* [9] (*Open Source Computer Vision Library*) es una biblioteca de código abierto diseñada para aplicaciones de visión por computadora y *machine learning*.

Su propósito principal es proporcionar herramientas para procesar y analizar imágenes o vídeos, permitiendo tareas como detección de objetos, reconocimiento facial, calibración de cámaras, seguimiento de movimiento o reconstrucción 3D. Funciona sobre matrices *NumPy* en *Python*, lo que facilita la manipulación de píxeles y operaciones matemáticas.

A continuación un ejemplo de como leer una imagen en *OpenCV*:

```
1 import cv2
2 # Cargar imagen en BGR (formato predeterminado de OpenCV)
3 imagen = cv2.imread('ruta/a/imagen.jpg')
4 # Convertir a escala de grises
5 gris = cv2.cvtColor(imagen, cv2.COLOR_BGR2GRAY)
6 # Guardar resultado
7 cv2.imwrite('imagen_gris.jpg', gris)
```

#### 4.3. PyTorch

*PyTorch* [10] es un *framework* de aprendizaje automático de código abierto especializado en redes neuronales y computación tensorial. *PyTorch* está diseñado para entrenar y desplegar modelos de *deep learning*, con énfasis en flexibilidad y velocidad mediante el uso de tensores y grafos computacionales dinámicos. Su arquitectura permite la diferenciación automática (autograd) y la ejecución en *GPUs*, lo que lo hace ideal para investigación y producción en visión por computador moderna.

A continuación un ejemplo de definición de una red neuronal de tipo *autoencoder* creada para una de las prácticas de una asignatura del máster en ciencia de datos:

```
1 class Encoder(torch.nn.Module):
2     def __init__(self, latent_dim):
3         super(Encoder, self).__init__()
4         self.encoder = torch.nn.Sequential(
```

```

5         torch.nn.Linear(in_features=28 * 28, out_features=100),  #
6             [B, 100]
7             torch.nn.ReLU(),  # [B, 100]
8             torch.nn.Linear(in_features=100, out_features=50),  # [B,
9                 50]
10            torch.nn.ReLU(),  # [B, 50]
11            torch.nn.Linear(in_features=50, out_features=latent_dim),
12             # [B, latent_dim]
13             )
14
15     def forward(self, x):  # x: [B, 1, 28, 28]
16         x = x.view(x.size(0), -1)  # [B, 1*28*28]
17         z = self.encoder(x)  # z: [B, latent_dim]
18         return z
19
20
21 class Decoder(torch.nn.Module):
22     def __init__(self, latent_dim):
23         super(Decoder, self).__init__()
24         self.decoder = torch.nn.Sequential(
25             torch.nn.Linear(in_features=latent_dim, out_features=50),
26             # [B, 50]
27             torch.nn.ReLU(),  # [B, 50]
28             torch.nn.Linear(in_features=50, out_features=100),  # [B,
29                 100]
30             torch.nn.ReLU(),  # [B, 100]
31             torch.nn.Linear(in_features=100, out_features=28 * 28),  # [
32                 B, 28*28]
33             torch.nn.Sigmoid(),  # [B, 28*28]
34             )
35
36     def forward(self, z):  # z: [B, latent_dim]
37         x_rec = self.decoder(z)  # x_rec: [B, 1*28*28]
38         # x_rec = self.decoder(z) # x_rec: [B, 1, 28, 28]
39         x_rec = x_rec.view(x_rec.size(0), 1, 28, 28)  # x_rec: [B, 1,
40             28, 28]
41         return x_rec
42
43
44 class Autoencoder(torch.nn.Module):
45     def __init__(self, latent_dim):
46         super(Autoencoder, self).__init__()
47         self.encoder = Encoder(latent_dim)
48         self.decoder = Decoder(latent_dim)
49
50     def forward(self, x):  # x: [B, 1, 28, 28]
51         z = self.encoder(x)  # z: [B, latent_dim]
52         x_rec = self.decoder(z)  # x_rec: [B, 1, 28, 28]
53         return x_rec

```

#### 4.4. Integración

Cada uno de los componentes aquí descritos son complejos sistemas basados en ingeniería, matemáticas, electrónica, física, etc. Todas estas componentes forman parte de un flujo de trabajo (fig 12) por el cual se extrae información con dispositivos de captura de imágenes, se almacena la información en discos duros, se preprocesa con librerías como *OpenCV*, se prepara el *dataset* de información, se define una arquitectura de red neuronal (si es esa la solución a utilizar), se entrena el modelo en computadores con *GPUs*, se prueba y valida y se despliega mediante soluciones tales como son servicios *web* o dispositivos embebidos.

El proceso puede variar mucho, pero se ha definido un flujo sencillo que integra todos los componentes *software* y *hardware* con un único propósito.

### 5. Aplicaciones actuales

Algunas aplicaciones actuales de todos estos tipos de sistemas de visión por computador son:

- **Conducción autónoma:** Se utilizan cámaras y sensores para detectar peatones, señales de tráfico, carriles y obstáculos en tiempo real, además de sistemas de estacionamiento automático, alertas de colisión o detección de fatiga mediante el seguimiento ocular.
- **Detección de anomalías:** Monitoreo de cámaras en tiempo real para identificar comportamientos sospechosos o intrusos.
- **Salud:** Análisis de radiografías, resonancias magnéticas o tomografías para detectar tumores, fracturas o enfermedades. Cirugía asistida guiada por robots quirúrgicos para mayor precisión que utilizan sistemas de visión por computador.
- **Agricultura:** En monitoreo de cultivos con drones o satélites que analizan el estado de los cultivos, detectan plagas o sequías.

Los sistemas de visión por computador están en todos lados y se utilizan día a día en multitud de ejemplos. Los ya mencionados son sólo unos pocos de los muchos tipos de aplicaciones que tienen este tipo de técnicas.



Figura 12: Ejemplo de integración entre componentes *software* y componentes *hardware* para un sistema de visión por computador.

## 6. Aplicación real: Detección de anomalías con *Deep Learning*

Los métodos de *deep learning* han revolucionado la detección de anomalías mediante enfoques no supervisados. Entre las técnicas más destacadas se encuentran los *Autoencoders Variacionales* (VAEs), las Redes Generativas Adversarias (GANs) y los *Autoencoders* convencionales, cada uno con mecanismos particulares para identificar patrones atípicos.

Los *Autoencoders Variacionales* (**VAE**) se fundamentan en un enfoque probabilístico donde el modelo aprende a codificar datos en distribuciones latentes en lugar de representaciones fijas. Esta característica permite no solo reconstruir entradas sino generar nuevas muestras coherentes. En detección de anomalías, se entrena exclusivamente con datos normales, estableciendo un umbral probabilístico: las muestras que presenten desviaciones significativas en su reconstrucción o baja probabilidad en el espacio latente se clasifican como anómalas. Su aplicación abarca desde la identificación de defectos en piezas industriales hasta el diagnóstico precoz de anomalías en imágenes de resonancia magnética [11].

Por otro lado, las **Redes Generativas Adversarias (GANs)** emplean una arquitectura dual donde un generador sintetiza datos y un discriminador los evalúa. Para detección de anomalías, variantes como *AnoGAN* comparan muestras reales con reconstrucciones generadas, asignando puntuaciones de anomalía basadas en discrepancias residuales y respuestas del discriminador.

Los *Autoencoders* tradicionales ofrecen un enfoque más directo, comprimiendo y reconstruyendo datos mediante capas neuronales simétricas. Al entrenarse únicamente con ejemplos normales, presentan altos errores de reconstrucción ante entradas atípicas.

## 7. Adquisición de imágenes

Se toma el ejemplo de adquisición del *dataset MVTec AD* [12], cuyas imágenes fueron obtenidas en un entorno de inspección industrial y están diseñadas para evaluar métodos de detección de anomalías sin supervisión. La captura de dichas imágenes tomó los siguientes pasos:

- **Captura de imágenes de objetos y texturas reales:** Se tomaron imágenes de 10 objetos diferentes y 5 tipos de texturas utilizando cámaras de alta resolución. Los objetos incluyen elementos industriales comunes, mientras que las texturas representan materiales de uso frecuente.
- **Imágenes sin defectos (para entrenamiento):** Se recopilaron imágenes de objetos y texturas en su estado normal (sin defectos), que sirven como referencia para modelos de detección de anomalías.

- **Imágenes con anomalías (para prueba):** Se generaron intencionalmente más de 70 tipos de defectos que incluyen araños, abolladuras, contaminación, cambios estructurales, etc. Estos defectos imitan los problemas reales que pueden ocurrir en la producción industrial.

Todas las imágenes fueron obtenidas utilizando un sensor **RGB** industrial de alta resolución de  $2048x2048$  píxeles en combinación con dos objetivos telecentrífugos bilaterales.

## 8. Preprocesamiento

Las imágenes fueron capturadas con diferentes niveles de ampliación ( $1 : 5$  y  $1 : 1$ ), recortadas para tener un tamaño adecuado, y sus resoluciones están entre  $700x700$  y  $1024x1024$  píxeles.

Se incluyen imágenes en escala de grises de objetos específicos, como rejillas, tornillos y cremalleras. La iluminación de las imágenes fue controlada rigurosamente, aunque en algunos casos se modificó intencionalmente para introducir más variabilidad en las condiciones.

Además, se proporciona información detallada sobre las áreas defectuosas de las imágenes, anotando manualmente casi 1900 regiones defectuosas con precisión de píxeles. Se muestra en la imagen de la figura 13

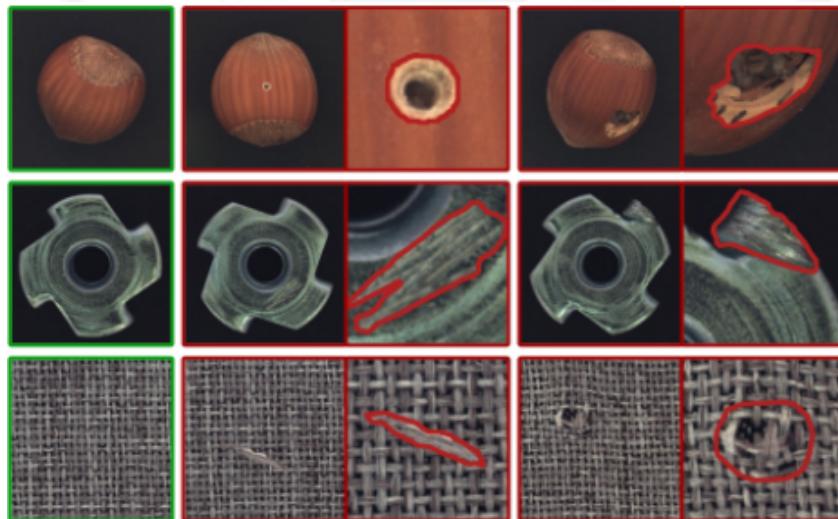


Figura 13: Dos objetos y una textura del conjunto de datos *MVTec AD* donde se muestran imágenes anómalias y normales.

## 9. Segmentación

La segmentación de imágenes utilizando algunos de los modelos descritos puede tomar distintos enfoques. Lo requerido en esta segmentación para este problema es obtener la parte anómala de la imagen.

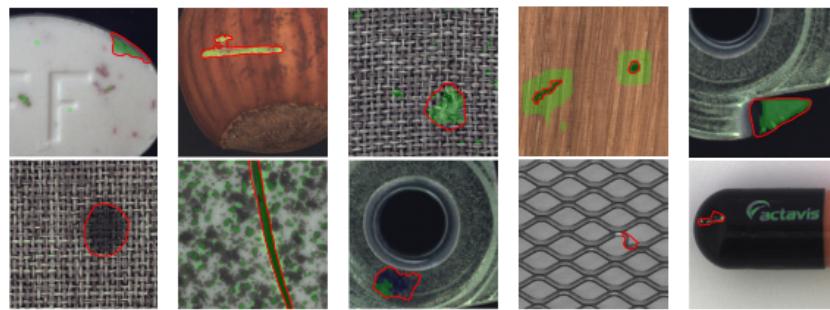


Figura 14: Segmentación de anomalías en imágenes. La primera imagen es segmentada por *AnoGAN* y la segunda por *Autoencoders* tradicionales

Los *Autoencoders*, por ejemplo, pueden utilizar [12] una arquitectura convolucional que procesa las imágenes completas cuando se trata de objetos, trabajando con resoluciones de  $256 \times 256$  píxeles. Sin embargo, cuando analizan texturas, toman un enfoque más detallado: dividen la imagen en parches más pequeños de  $128 \times 128$  píxeles y los analizan con un solapamiento, moviéndose cada 30 píxeles para cubrir toda la imagen. La segmentación final se obtiene comparando la imagen reconstruida con la original, ya sea mediante una simple diferencia píxel a píxel o utilizando un índice más sofisticado de similitud estructural.

El método *AnoGAN* toma un enfoque diferente, utilizando redes generativas adversarias. Este sistema se entrena exclusivamente con imágenes sin defectos y, cuando analiza una nueva imagen, intenta reproducirla utilizando su espacio latente. Las anomalías se detectan comparando la imagen generada con la original. Para manejar diferentes tipos de datos, redimensiona los objetos a  $128 \times 128$  píxeles, mientras que para las texturas analiza la imagen por parches del mismo tamaño.

Ejemplos de este tipo se pueden observar en la figura 14.

## 10. Extracción de características

Este tipo de redes neuronales aceptan como entrada las imágenes en crudo como un vector de  $N \times M$  características, siendo cada una de ellas un valor de un píxel. Redes como las *VAEs* representan estas características en el **espacio latente**. Este se define como un espacio abstracto multidimensional que codifica una representación interna significativa

de los sucesos observados externamente. Las muestras que son similares en el espacio original se sitúan cerca unas de otras en el espacio latente. De esta forma se extraen y resumen aquellas características más relevantes.

A partir de estas características es posible reconstruir la imagen de nuevo, de forma que en el proceso, aquellas patrones anómalos serán reconstruidos con un error superior a la media de error de reconstrucción del resto de imágenes.

## 11. Clasificación

La detección de anomalías se fundamenta en la capacidad de identificar discrepancias significativas entre la imagen original y la imagen reconstruida o procesada por el modelo. Un enfoque común es la **umbralización del error de reconstrucción**, en el que se mide la diferencia, utilizando métricas como el error cuadrático medio o la diferencia absoluta, entre la imagen original y la generada por el modelo. Al establecer un umbral de error (definido a partir de las características estadísticas del conjunto de entrenamiento) se puede clasificar una muestra como anómala si el error de reconstrucción supera dicho valor. Este método, sencillo y eficaz, es particularmente útil cuando la distribución de errores de reconstrucción en las imágenes normales es lo suficientemente homogénea.

Otro enfoque importante se basa en el análisis del **espacio latente**. Durante el entrenamiento, modelos como los Autoencoders Variacionales (VAE) aprenden a codificar las imágenes en un espacio multidimensional donde se representan de forma compacta las características esenciales de los datos. Dado que el entrenamiento se realiza únicamente con ejemplos normales, las representaciones de imágenes defectuosas suelen ubicarse alejadas de los clusters formados por los datos normales. Esta diferencia en la distribución permite utilizar técnicas de clustering o análisis de distancia para identificar y clasificar de forma robusta las anomalías, ofreciendo una alternativa efectiva a la mera umbralización del error de reconstrucción.

Por otro lado, las **Redes Generativas Adversarias (GANs)** introducen una estrategia de clasificación basada en la interacción entre el generador y el discriminador. En este esquema, el generador intenta reproducir la imagen de entrada a partir de su espacio latente, mientras que el discriminador evalúa la similitud entre la imagen original y la generada. Una discrepancia significativa, ya sea en términos de la diferencia de contenido o de la respuesta del discriminador, se interpreta como indicativo de una anomalía. Este enfoque, popularizado en técnicas como *AnoGAN*, permite asignar una puntuación de anomalía que facilita la toma de decisiones en la fase de clasificación.

La evaluación del desempeño de estos métodos se realiza mediante diversas métricas, tales como la *accuracy*, *precision*, *recall* y la puntuación *F1-Score*. Asimismo, el análisis de la curva ROC y el cálculo del AUC proporcionan una visión global de la capacidad del sistema para distinguir entre muestras normales y anómalas. Estas herramientas de evaluación son fundamentales para afinar los parámetros del modelo y garantizar una

detección precisa en entornos reales.



## 12. Bibliografía

- [1] OpenCV Courses, *A Brief History of Computer Vision*, 2023. URL: <https://opencv.courses/blog/a-brief-history-of-computer-vision/>.
- [2] D. A. Lienhard, “David H. Hubel and Torsten N. Wiesel’s Research on Optical Development in Kittens,” *Embryo Project Encyclopedia*, oct. de 2017, ISSN: 1940-5030. URL: <https://hdl.handle.net/10776/12995>.
- [3] L. Roberts, *Machine Perception of Three-Dimensional Solids*. ene. de 1963, ISBN: 0-8240-4427-4.
- [4] D. Marr, *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*. San Francisco: W. H. Freeman y Company, 1982, ISBN: 978-0716715672.
- [5] J. Canny, “A Computational Approach to Edge Detection,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, n.º 6, págs. 679-698, 1986. DOI: 10.1109/TPAMI.1986.4767851.
- [6] D. G. Lowe, “Distinctive Image Features from Scale-Invariant Keypoints,” en *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, IEEE, 2004, págs. 1150-1157. DOI: 10.1109/ICCV.2004.223914.
- [7] P. Viola y M. Jones, “Rapid Object Detection using a Boosted Cascade of Simple Features,” en *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE Computer Society, 2001, págs. 511-518, ISBN: 0-7695-1272-0. URL: <http://dblp.uni-trier.de/db/conf/cvpr/cvpr2001-1.html#ViolaJ01>.
- [8] A. Krizhevsky, I. Sutskever y G. E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks,” en *Proceedings of the 25th International Conference on Neural Information Processing Systems (NIPS)*, Curran Associates, Inc., 2012, págs. 1097-1105. DOI: 10.1145/3065386. URL: <https://dl.acm.org/doi/10.1145/3065386>.
- [9] Itseez, *Open Source Computer Vision Library*, <https://github.com/itseez/opencv>, 2015.
- [10] A. Paszke et al., “PyTorch: An Imperative Style, High-Performance Deep Learning Library,” en *Advances in Neural Information Processing Systems 32*, Curran Associates, Inc., 2019, págs. 8024-8035. URL: <http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf>.
- [11] H. H. Nguyen, C. N. Nguyen, X. T. Dao, Q. T. Duong, D. P. T. Kim y M.-T. Pham, *Variational Autoencoder for Anomaly Detection: A Comparative Study*, 2024. arXiv: 2408.13561 [cs.CV]. URL: <https://arxiv.org/abs/2408.13561>.

- [12] P. Bergmann, M. Fauser, D. Sattlegger y C. Steger, “MVTec AD — A Comprehensive Real-World Dataset for Unsupervised Anomaly Detection,” en *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2019, págs. 0-0. URL: [https://www.mvttec.com/fileadmin/Redaktion/mvttec.com/company/research/datasets/mvttec\\_ad.pdf](https://www.mvttec.com/fileadmin/Redaktion/mvttec.com/company/research/datasets/mvttec_ad.pdf).

