



Datamecum

Modelos fundacionales en visión artificial

Tema 6: Otras tareas de visión

Día 2 del curso

Web del curso	datamecum.sergiovillanuevalopez.com
Práctica	Cuaderno 6 en Colab (6_Otras_Tareas.ipynb)
Profesor	Sergio Villanueva · Datamecum

Otras tareas fundacionales de visión artificial

Seis tareas más, cada una con su modelo fundacional.

Estimación de pose

Identificar y localizar puntos clave del cuerpo humano o de objetos en imágenes y vídeos.

Superresolución

Mejorar la calidad y resolución de imágenes o vídeos, a menudo a partir de datos de baja resolución.

Estimación de profundidad

Calcular la distancia de los objetos en una escena respecto a la cámara, creando mapas de profundidad.

OCR moderno

Extraer texto de imágenes y documentos, transformándolo en datos editables y buscables.

Eliminación de fondo

Segmentar y eliminar automáticamente el fondo de una imagen, aislando el objeto principal.

Correspondencias entre imágenes

Encontrar correspondencias o similitudes entre dos o más imágenes para tareas como la localización o el reconocimiento.

Estimación de pose humana: de píxeles a esqueletos

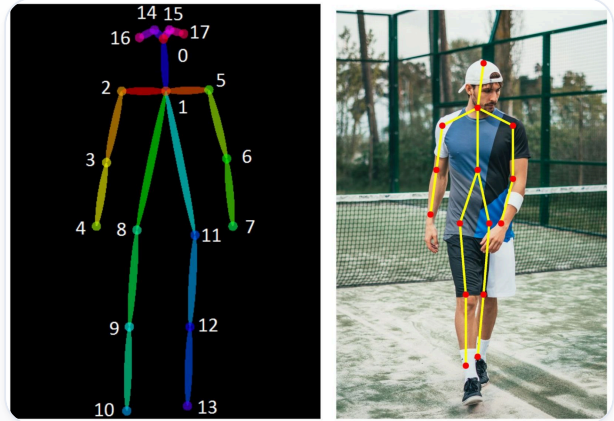
Estructurando el cuerpo humano en coordenadas digitales.

¿Qué es? Tarea de visión que localiza articulaciones clave (keypoints) para modelar la postura y el movimiento.

El estándar (COCO-17): se buscan típicamente 17 puntos.

- **Cabeza:** nariz, ojos (2), orejas (2).
- **Tren superior:** hombros (2), codos (2), muñecas (2).
- **Tren inferior:** caderas (2), rodillas (2), tobillos (2).

El output: una lista de coordenadas (x, y) más un score de confianza por punto.



Transformamos una matriz de píxeles ininteligible en una estructura geométrica analizable (skeleton).

Enfoques de modelado: top-down vs. bottom-up

Trade-off entre precisión y velocidad en multitudes.

Top-down (alta precisión)

Proceso: primero detecta personas (cajas), luego estima la pose dentro de cada una.

Ejemplos: HRNet, ViTPose.

- Pros: máxima precisión, robustez a distintas escalas.
- Contrás: lento en grandes aglomeraciones, con coste lineal en el número de personas.

Bottom-up (alta velocidad)

Proceso: detecta todos los puntos clave de la imagen a la vez, luego los agrupa en personas.

Ejemplo: OpenPose.

- Pros: velocidad constante, independiente del número de personas.
- Contrás: menos preciso, el agrupamiento de puntos es complejo.

Single-stage (el equilibrio moderno)

Proceso: redes como YOLO-Pose predicen cajas y puntos simultáneamente en una sola pasada (end-to-end).

- Ventaja: velocidad en tiempo real con precisión competitiva, ideal para dispositivos edge.

Modelos destacados

Herramientas prácticas para cada escenario.

1

MediaPipe (Google)

Funciona bien en CPU y en el móvil. Pensado para fitness, filtros AR y escenas con una o pocas personas cerca de la cámara. Tiene también modelos para manos y cara.

2

YOLO-Pose (Ultralytics)

YOLOv8, YOLO11 y YOLO26 en versión pose: varias personas en una pasada, buen equilibrio entre velocidad y precisión y fáciles de reentrenar. Licencia AGPL-3.0, como el resto de Ultralytics (tema 1).

3

OpenPose (CMU)

El pionero académico que estableció las bases. Aunque influyente, hoy en día es computacionalmente pesado en comparación con opciones más modernas.

4

HRNet

Prioridad en precisión extrema. Cuando la exactitud es crítica (análisis médico, biomecánico) y la velocidad de inferencia es secundaria.

5

ViTPose (el del cuaderno)

Top-down sobre un ViT sencillo: en el cuaderno, RT-DETR encuentra a las personas y ViTPose les pone los 17 puntos. Licencia Apache 2.0.

¿Cómo medimos la calidad? La métrica OKS

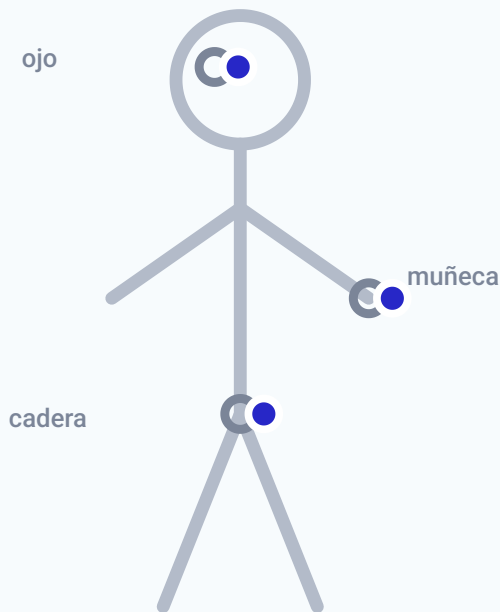
OKS (Object Keypoint Similarity) hace con los esqueletos lo que IoU hace con las cajas: convierte la distancia entre cada punto predicho y el real en una puntuación de 0 a 1, corregida por el tamaño de la persona y por lo difícil que es marcar esa articulación.

Por eso **el mismo error no penaliza igual**: 5 píxeles de error en el ojo, que los anotadores marcan con mucha precisión, cuestan bastante más que 5 píxeles en la cadera, cuya posición exacta es más difícil de fijar.



El mismo error no penaliza igual

La tolerancia depende de la articulación y de la escala de la persona (la raíz de su área).
Mueve las dos cosas y mira cuándo la pose deja de contar como acierto.



El muñeco mide siempre lo mismo y representa a la persona entera: si la persona encoge, el mismo error en píxeles se ve mayor sobre ella. Las tolerancias son las constantes de COCO.

Cuenta como acierto

OKS del esqueleto

Umbral de acierto

0,92

0,50

Ojo σ 0,025

0,80

Muñeca σ 0,062

0,96

Cadera σ 0,107

0,99

Con escala de 150 px, 5 px dejan el ojo en 0,80 y la cadera en 0,99. El ojo pierde medio punto a partir de 9 px.

Esto explica el fallo más habitual en planta: el modelo va bien con la persona cerca y se desmorona con la persona al fondo, aunque el error en píxeles sea el mismo. Las dos salidas son acercar la cámara o trabajar en top-down, detectando primero a cada persona y recortándola para que llegue grande al modelo de pose.

Esta página recoge el estado con 5 px de error y escala de 150 px. La escala de COCO es la raíz del área de la persona, y el OKS de la pantalla promedia solo tres articulaciones para que se vea el efecto. OKS aplica a cada punto $\exp(-d^2 / (2 s^2 k^2))$, donde d es el error en píxeles, s la escala de la persona y $k = 2\sigma$ una constante propia de cada articulación medida sobre anotaciones humanas. Los ojos tienen $\sigma = 0,025$ y la cadera $\sigma = 0,107$, así que la cadera tolera más de cuatro veces el error antes de penalizar lo mismo. La escala entra dividiendo, de modo que la tolerancia en píxeles es proporcional al tamaño de la persona: en una persona de 60 px el ojo pierde medio punto a partir de 4 px, y en una de 400 px aguanta hasta 24 px. El OKS del esqueleto es la media de los puntos visibles, y sustituye al IoU cuando se calcula AP para pose: COCO promedia AP con umbrales de 0,50 a 0,95, así que el 0,50 de esta pantalla es el criterio más indulgente de los diez.

Cinco píxeles no son cinco píxeles

La tolerancia de cada articulación sale de cómo de acuerdo se ponen los anotadores humanos.

Dos articulaciones fallan cinco píxeles. ¿Penaliza OKS siempre igual?

- ☐ A Sí, solo cuenta la distancia en píxeles.
- ☐ B No, depende de escala y articulación.
- ☐ C No, solo cuenta cuántas personas hay.

Respuesta B. Exacto: normaliza por tamaño y usa una tolerancia específica por keypoint.

Las constantes por articulación se estimaron midiendo la dispersión entre anotadores. Donde las personas coinciden mucho, como los ojos, el modelo tiene menos margen.

Impacto real y desafíos técnicos

Dónde genera valor la estimación de pose.

Deporte y fitness

- **Entrenador IA:** conteo preciso de repeticiones (sentadillas, flexiones...).
- **Análisis de técnica:** medición de ángulos corporales para prevenir lesiones.
- Por ejemplo, Peloton Guide cuenta repeticiones con una cámara.

Seguridad industrial y ergonomía

- **Detección de posturas peligrosas:** identificar movimientos de riesgo al levantar peso.
- **«Hombre caído»:** alertas de seguridad en tiempo real en entornos peligrosos.
- **Optimización:** mejora de la eficiencia en líneas de montaje.

Salud y rehabilitación

- **Fisioterapia remota:** validación de ejercicios en casa sin supervisión presencial.
- **Análisis de marcha:** detección temprana de patologías neurológicas o musculoesqueléticas.

El gran reto: oclusiones y reconstrucción 3D

El problema: la estimación de pose se complica cuando partes del cuerpo se ocultan entre sí o tras otros objetos, y se pierde información.

La solución: modelos avanzados emplean memoria temporal para rastrear y predecir oclusiones, o realizan inferencia 3D para «imaginar» las partes ocultas basándose en la anatomía y el contexto.



OCR moderno: de leer caracteres a entender documentos

La evolución de Tesseract a los transformers (TrOCR).

El pasado: enfoque clásico (Tesseract)

Metodología: hasta la versión 3 segmentaba caracteres con reglas y contornos; desde la 4 (2018) reconoce líneas con una red LSTM, pero sigue necesitando texto limpio y horizontal.

Limitaciones comunes:

- Falla con texto curvo, distorsionado o en perspectivas inusuales.
- Poca robustez ante fuentes manuscritas complejas o variaciones tipográficas.
- Sensible al ruido de fondo, baja resolución o iluminación deficiente.
- Requiere preprocesamiento intensivo (normalización, binarización).

El presente: transformers para OCR (TrOCR y VLM)

Arquitectura end-to-end: utiliza Vision Transformers (ViT) para codificar la imagen de forma global y un decoder de texto (en TrOCR, inicializado desde RoBERTa) genera la secuencia de caracteres.

Lo que cambia: lee la línea entera de una vez, con el contexto de las letras vecinas, en lugar de reconocer caracteres sueltos.

Estado del arte (SOTA):

- **TrOCR (Microsoft):** un referente específico para tareas de OCR.
- **Qwen3-VL y Qwen3.5:** VLM que leen y además razonan sobre el texto, por ejemplo devolviendo en JSON los campos de una factura.

Herramientas prácticas de OCR

Implementación rápida con Hugging Face y EasyOCR.

EasyOCR: el comodín versátil

Una opción todo terreno para cualquier proyecto. Es ligera, soporta más de 80 idiomas y funciona eficientemente en CPU. Ideal para prototipos rápidos y escenarios donde no se dispone de GPU.

TrOCR (Hugging Face): precisión avanzada

Muy preciso en texto impreso y en manuscrito, según la variante. Lee una sola línea ya recortada, así que necesita un detector de texto delante. Va mucho mejor con GPU.

VLM (Qwen, Gemini, GPT): documentos estructurados

Perfectos para casos de uso complejos como el procesamiento de facturas o tablas. Permiten extraer información estructurada, por ejemplo JSON, en lugar de solo texto plano, integrando visión y lenguaje para un razonamiento avanzado.

Lo que devuelve cada generación de OCR

La misma foto de una placa, leída de tres formas. Cambia la herramienta y mira qué recibe tu programa.



Serigrafía de una placa real, con texto pequeño, girado por el brillo del cobre y mezclado con los componentes. Las cajas están medidas a mano para enseñar el formato de salida.

```
C1 C5 R6 C4 PULSE
T1 R3 MCP6004
IC2 R4 R11 R10 R5 R9
J1 VCC EN VOUT GND
R1 R2 R7 R8
Place finger SENSOR
Easy Pulse
www.Embedded-Lab.com
```

Sale una tira de caracteres y poco más. El orden lo inventa el recorrido por líneas, así que «VCC EN VOUT GND» y las referencias de las resistencias se mezclan sin que puedas saber qué va con qué.

La pregunta útil es qué necesita tu programa, más que qué herramienta lee mejor. Para contar si la serigrafía está impresa basta la primera; para comprobar que cada referencia está junto a su componente hacen falta coordenadas; para rellenar un parte de recepción con el modelo y el fabricante, el tercero te ahorra el pegamento.

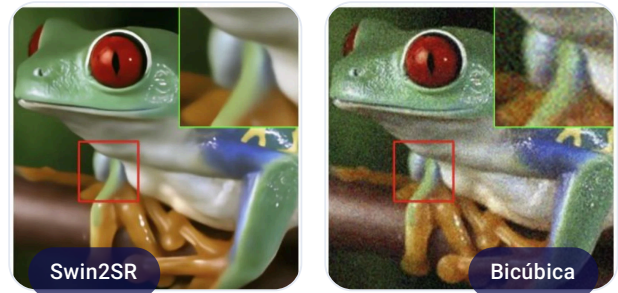
Las cajas de esta pantalla están medidas a mano sobre la foto para ilustrar el formato de salida, no las ha producido un OCR. Tres cosas que se ven en esta imagen y que aparecen en cualquier planta: el texto de la serigrafía es de bajo contraste sobre verde y el brillo del cobre lo rompe, hay texto a distintas escalas en la misma foto, y varias referencias quedan parcialmente tapadas por los componentes. Por eso la resolución de captura manda más que el modelo: con menos de veinte píxeles de altura por carácter, ningún OCR actual lee de forma fiable. Sobre latencia y coste, Tesseract corre en CPU en decenas de milisegundos, un modelo de detección más reconocimiento ronda las décimas de segundo en GPU, y un VLM se va a varios segundos por imagen, así que en línea suele combinarse: detección rápida siempre y VLM solo sobre las piezas dudosas.

Superresolución: restaurando el detalle perdido

El uso de deep learning para generar detalles plausibles de alta frecuencia a partir de una imagen de baja resolución, superando los límites de la interpolación tradicional (bilineal, bicúbica...).

El modelo del cuaderno: Swin2SR (2022).

- Basado en Swin Transformers V2, lo que le permite procesar imágenes de manera eficiente y global.
- Entrenado para escalar la imagen y también para eliminar artefactos de compresión JPEG y ruido.
- **Lo que aporta:** recupera y sintetiza texturas finas (como pelo, tela o detalles faciales) que los métodos clásicos dejarían borrosas o artificiales.



Mira el recuadro rojo: la textura fina de la derecha la sintetiza el modelo.

Nota importante: la superresolución no es magia. El modelo «imagina» detalles a partir de los patrones que vio al entrenar; no reconstruye información que nunca existió.

Aplicaciones reales y limitaciones

Dónde aplicar superresolución en industria.

Restauración de archivos

- Digitalización de fotos antiguas o documentos históricos dañados.
- Mejora de contenido multimedia antiguo (remasterización).

Medicina e imagen satelital

- Mejorar la legibilidad de escáneres médicos.
- Zoom en imágenes aéreas o satelitales.

E-commerce y marketing

- Mejorar fotos de productos de baja calidad enviadas por usuarios o proveedores.
- Lograr una apariencia profesional en la web para el catálogo de productos.

Riesgo crítico: «alucinaciones». El modelo puede inventar un tumor o una mancha. Nunca usar para diagnóstico crítico sin supervisión.

Detalle generado no es evidencia

Lo que aparece al ampliar puede venir del modelo, no del sensor.

Una superresolución muestra una mancha nueva en una imagen médica. ¿Qué concluyes?

- ☐ A La mancha estaba en el original.
- ☐ B Hay que validarla contra el original y el proceso clínico.
- ☐ C Una mayor resolución siempre elimina el error.

Respuesta B. Exacto: la mejora visual no convierte el detalle generado en evidencia.

La superresolución sintetiza textura plausible a partir de patrones aprendidos. En un contexto diagnóstico, cualquier estructura nueva debe comprobarse sobre la imagen original.

Eliminación de fondo profesional

El reto del «alpha matting» y el modelo RMBG.

El problema: bordes difusos

Segmentar objetos sólidos es fácil. Pero elementos como el pelo, el pelaje o el vidrio transparente son extremadamente difíciles de separar debido a sus bordes difusos y semitransparentes.

La solución: RMBG-1.4 (BRIA AI)

Modelo pensado para alpha matting: genera un canal alfa con transparencia gradual en los bordes en lugar de una máscara binaria, así que el recorte encaja en cualquier fondo sin efecto «pegatina». Su licencia no permite uso comercial sin acuerdo con BRIA.

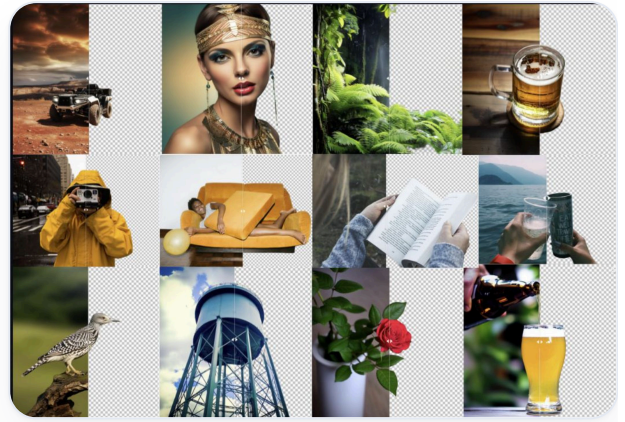


Image matching: encontrando parejas visuales

De los algoritmos clásicos (SIFT, ORB...) a las redes.

¿Qué es? Encontrar puntos idénticos en dos fotos de la misma escena tomadas desde ángulos diferentes, a pesar de cambios en la perspectiva o la iluminación. Es la base de la reconstrucción 3D, la realidad aumentada y la localización de robots.

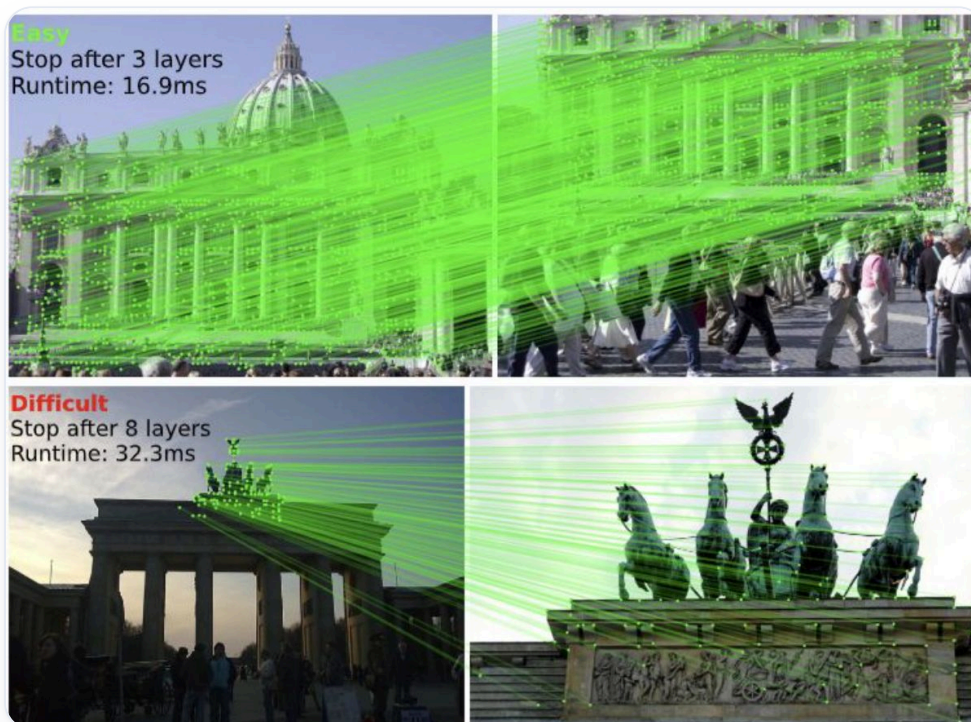
La nueva era: deep learning

Detector (SuperPoint, DISK)

Una red que localiza los puntos más distintivos de la imagen (esquinas, texturas) y describe cada uno con un vector. Aguanta mejor que SIFT los cambios fuertes de luz y de punto de vista.

Matcher (LightGlue)

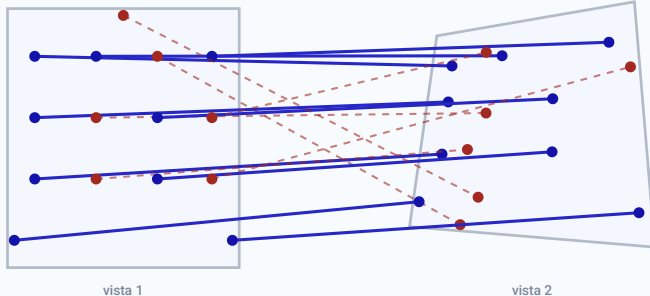
Un Transformer que compara los puntos de las dos imágenes, decide cuáles se corresponden y descarta los que no tienen pareja. Los errores geométricos que se le escapan se filtran después con RANSAC, como en la pantalla siguiente.



Dónde se nota: con cambios de luz (de día a noche), grandes variaciones de perspectiva y oclusiones, justo donde los métodos clásicos fallan más.

La mitad de las parejas sobran

Un matcher devuelve más parejas de las que valen. El filtro que las separa no mira el aspecto, mira la geometría: mueve el umbral y comprueba cuáles encajan con la escena.



Esquema con una homografía real: la misma fachada vista desde otro sitio. Las parejas y su error se calculan en el navegador.

Estimación sólida

Parejas aceptadas

9

Descartadas

6

9 parejas coherentes y 6 descartadas. Las descartadas son las que se parecían sin estar en el mismo sitio.

Este umbral es el que se pasa a `cv2.findHomography` con RANSAC, y es el mando que más cambia el resultado en fotogrametría y en SLAM. Demasiado bajo y te quedas sin puntos para calcular; demasiado alto y una ventana confundida con otra te tuerce la pose de la cámara.

Esta página recoge el estado con 2 px de umbral. El procedimiento real es RANSAC: se eligen cuatro parejas al azar, se calcula la homografía que implican, se cuenta cuántas de las demás quedan por debajo del umbral de reproyección, y se repite unos cientos de veces quedándose con la hipótesis que más apoyo reúne. Esta pantalla enseña el paso de recuento con la homografía correcta ya fijada, que es donde se ve el compromiso. Las cuatro parejas muy alejadas son los errores típicos del matching: dos ventanas iguales en fachadas repetidas, un reflejo y un punto en el cielo, donde no hay textura estable. Las dos parejas intermedias, a 4,5 y 7,5 px, son las que deciden si el umbral está bien puesto: entran al subirlo y arrastran error a la estimación. Con imágenes de verdad, el umbral se elige en píxeles de la imagen original, así que cambiar la resolución de captura obliga a revisarlo.

Reconstruyendo el mundo en 3D

Tres usos del matching.

Fotogrametría (SfM)

Al identificar y conectar puntos coincidentes en múltiples fotos de una escena, se puede reconstruir la geometría 3D del objeto y la posición de la cámara en cada toma.

SLAM (robótica)

Un robot utiliza el matching de características para comprender su posición dentro de un entorno y construir un mapa del mismo, incluso mientras se mueve y explora.

Panoramas (stitching)

Permite fusionar sin problemas varias imágenes superpuestas en una única foto de gran formato o una vista de 360 grados, eliminando duplicados y alineando las perspectivas.

Depth estimation: 3D desde una sola imagen

Profundidad con una sola cámara y una sola foto.

Depth Anything V2/V3

Un modelo fundacional con encoder DINOv2. La V2 se entrenó con 595.000 imágenes sintéticas etiquetadas y 62 millones de imágenes reales con pseudoetiquetas.

Capacidad

Genera un mapa de profundidad (heatmap) donde el color indica la distancia relativa, revelando la estructura 3D subyacente.

Robustez

Aguanta interiores, exteriores y escenas con mucho detalle. Cristales y espejos, que hacían fallar a la V1, los maneja mucho mejor gracias a los datos sintéticos; aun así, compruébalo con tus imágenes.



Una sola cámara y una sola imagen: el modelo deduce qué está cerca y qué está lejos.

Recortar una foto por distancia

El mapa de profundidad de la pantalla anterior, usado como filtro: todo lo que queda más lejos que el corte se aclara. No hacen falta clases ni etiquetas: basta un umbral sobre la cercanía.

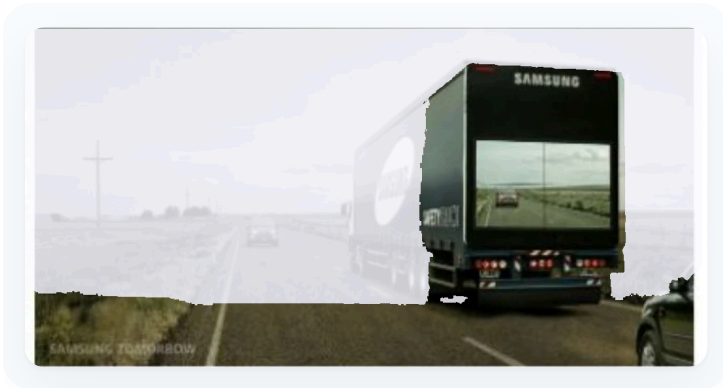


Imagen que se conserva

42 %

Queda la trasera del camión, y la pantalla que muestra otra carretera sigue entera: el modelo le da la distancia del camión aunque enseñe una carretera lejana.

Mapa precalculado, el de la pantalla anterior; el corte se aplica en tu navegador. La escala es relativa: 100 no equivale a ninguna distancia en metros.

Esta página recoge el corte en 100 de 255, que conserva el 42 % de la imagen: la trasera del camión, el asfalto cercano y el coche de la derecha. El costado del camión se aleja de la cámara y queda fuera. La pantalla trasera, que muestra la carretera de delante, se conserva entera porque el modelo mide a qué distancia está la superficie de la pantalla; la carretera que aparece en ella no cuenta. Con el corte en 30 solo desaparece el cielo; por encima de 135 el camión entero queda al otro lado.

Aplicaciones de la profundidad

¿Para qué sirve un mapa de profundidad?

Efecto retrato (bokeh)

Desenfocar el fondo sintéticamente, creando imágenes con una profundidad de campo profesional y destacando el sujeto principal.

Robótica y navegación

Permitir que robots y vehículos autónomos detecten obstáculos y comprendan su entorno 3D, utilizando solo una cámara de bajo coste.

Generación de nubes de puntos

Convertir una foto plana en una estructura 3D navegable (malla o nube de puntos) al combinar la imagen RGB con su mapa de profundidad.

Recorte por distancia

Un umbral sobre el mapa elimina lo que queda más lejos (o más cerca) de lo que interesa, sin clases ni etiquetas.

Profundidad relativa vs. métrica. Estos modelos suelen estimar profundidad relativa (mapas de calor). Para obtener metros reales se necesita calibración de cámara o un modelo de profundidad métrica; Depth Anything V2 tiene variantes métricas para interior y exterior.

Relativa no es métrica

Un mapa normalizado ordena distancias, pero no las mide.

Necesitas saber si un obstáculo está a 1,5 metros. ¿Basta un mapa relativo normalizado?

- ☐ A Sí, porque el color ya representa metros.
- ☐ B Sí, si la imagen tiene buena resolución.
- ☐ C No, hace falta escala métrica o calibración.

Respuesta C. Exacto: necesitas un modelo métrico, calibración u otra referencia de escala.

Un mapa relativo dice qué está más cerca que qué. Para responder «a cuántos metros» hacen falta calibración de cámara, una referencia conocida o un modelo entrenado para profundidad métrica.

Cuatro sectores, las tareas del curso

Los mismos modelos, combinados de otra manera, resuelven problemas muy distintos. Lo que cambia de un sector a otro es qué hay que vigilar.

Vigilancia y seguridad

Detectar y seguir personas y vehículos, contar los que entran en una zona, avisar de una caída con la pose. RF-DETR con un seguidor (tema 1) y ViTPose (este tema).

Ojo: el Reglamento europeo de IA prohíbe desde febrero de 2025, salvo excepciones tasadas, la identificación biométrica remota en tiempo real en espacios públicos con fines policiales y el reconocimiento de emociones en el trabajo.

Robótica

Saber qué está cerca con una sola cámara, recortar el objeto que hay que coger, encontrar objetos nuevos por su nombre. Depth Anything (este tema), SAM 2 (tema 5) y Grounding DINO (tema 3).

Ojo: esta profundidad es relativa; para coger un objeto hacen falta metros.

Industria

Defectos sin ejemplos de defecto, conteo de piezas, lectura de lotes y etiquetas, medidas con máscaras. DINOv3 con banco de normalidad (tema 4), SAM (tema 5) y OCR (este tema).

Ojo: luz y encuadre estables, y un umbral fijado con piezas buenas.

Automoción

Vehículos, peatones y señales, la calzada separada del resto, distancias y matrículas. Detectores en tiempo real (tema 1), segmentación (tema 5), profundidad y OCR (este tema).

Ojo: tiempo real en el hardware del coche, y noche, lluvia y contraluz en los datos de validación.

Las prohibiciones del artículo 5 del Reglamento (UE) 2024/1689 se aplican desde el 2 de febrero de 2025. Entre ellas están la identificación biométrica remota en tiempo real en espacios de acceso público con fines policiales, salvo excepciones tasadas, y el reconocimiento de emociones en el lugar de trabajo y en centros educativos, salvo por motivos médicos o de seguridad. Contar personas sin identificarlas no entra en esas prohibiciones, pero sigue sujeto al RGPD. Para la profundidad métrica hace falta un modelo entrenado para ello, una cámara estéreo o calibración.

Las seis tareas, seis recetas

Lo mínimo para arrancar cada una, con el mismo código que usa el cuaderno. Elige la tarea.

```
# Pose: personas y sus 17 puntos
import torch
from PIL import Image
from transformers import AutoProcessor, RTDetForObjectDetection, VitPoseForPoseEstimation

image = Image.open("mi_imagen.jpg").convert("RGB")

# 1. Encontrar a las personas
det_processor = AutoProcessor.from_pretrained("PekingU/rt detr_r50vd_coco_o365")
detector = RTDetForObjectDetection.from_pretrained("PekingU/rt detr_r50vd_coco_o365")
with torch.no_grad():
    detections = detector(**det_processor(images=image, return_tensors="pt"))
found = det_processor.post_process_object_detection(
    detections, target_sizes=[(image.height, image.width)], threshold=0.5)[0]
person_id = [i for i, name in detector.config.id2label.items() if name == "person"][0]
boxes = found["boxes"][found["labels"] == person_id].numpy()
boxes[:, 2:] -= boxes[:, :2] # ViTPose quiere x, y, ancho, alto

# 2. Ponerles el esqueleto
pose_processor = AutoProcessor.from_pretrained("usyd-community/vitpose-plus-base")
pose_model = VitPoseForPoseEstimation.from_pretrained("usyd-community/vitpose-plus-base")
inputs = pose_processor(image, boxes=[boxes], return_tensors="pt")
inputs["dataset_index"] = torch.tensor([0]) # 0: los 17 puntos de COCO
with torch.no_grad():
    poses = pose_model(**inputs)
people = pose_processor.post_process_pose_estimation(poses, boxes=[boxes])[0]
print(len(people), people[0]["keypoints"].shape) # personas, (17, 2)
```

```
# OCR: todo el texto de la imagen, línea a línea
import torch
from PIL import Image
from transformers import AutoModelForImageTextToText, AutoProcessor

image = Image.open("mi_documento.jpg").convert("RGB")
processor = AutoProcessor.from_pretrained("stepfun-ai/GOT-OCR-2.0-hf")
model = AutoModelForImageTextToText.from_pretrained("stepfun-ai/GOT-OCR-2.0-hf", device_map="auto")

inputs = processor(image, return_tensors="pt").to(model.device, model.dtype)
with torch.no_grad():
    ids = model.generate(**inputs, do_sample=False, tokenizer=processor.tokenizer,
                        stop_strings=["<|im_end|>"], max_new_tokens=500)
print(processor.decode(ids[0, inputs["input_ids"].shape[1]:], skip_special_tokens=True))
```

```
# Superresolución: el doble de lado
import numpy as np
import torch
from PIL import Image
from transformers import AutoImageProcessor, Swin2SRForImageSuperResolution

image = Image.open("mi_imagen.jpg").convert("RGB")
processor = AutoImageProcessor.from_pretrained("caidas/swin2SR-classical-sr-x2-64")
model = Swin2SRForImageSuperResolution.from_pretrained("caidas/swin2SR-classical-sr-x2-64")

with torch.no_grad():
    output = model(**processor(image, return_tensors="pt"))
    pixels = output.reconstruction.squeeze().clamp(0, 1).permute(1, 2, 0).numpy()
    Image.fromarray((pixels * 255).astype(np.uint8)).save("ampliada.png")
```

```
# Fondo: las personas recortadas sobre fondo transparente
import numpy as np
import torch
from PIL import Image
from transformers import AutoProcessor, RTDetrForObjectDetection, Sam2Model, Sam2Processor

image = Image.open("mi_imagen.jpg").convert("RGB")

# 1. Encontrar a las personas
det_processor = AutoProcessor.from_pretrained("PekingU/rt detr_r50vd_coco_o365")
detector = RTDetrForObjectDetection.from_pretrained("PekingU/rt detr_r50vd_coco_o365")
with torch.no_grad():
    detections = detector(**det_processor(images=image, return_tensors="pt"))
    found = det_processor.post_process_object_detection(
        detections, target_sizes=[(image.height, image.width)], threshold=0.5)[0]
    boxes = [b.tolist() for b, l in zip(found["boxes"], found["labels"])]
    if detector.config.id2label[l.item()] == "person"]

# 2. Convertir cada caja en una silueta con SAM 2
sam_processor = Sam2Processor.from_pretrained("facebook/sam2.1-hiera-small")
sam = Sam2Model.from_pretrained("facebook/sam2.1-hiera-small")
inputs = sam_processor(images=image, input_boxes=[boxes], return_tensors="pt")
with torch.no_grad():
    outputs = sam(**inputs)
    masks = sam_processor.post_process_masks(outputs.pred_masks, inputs["original_sizes"])[0]
    best = outputs.iou_scores[0].argmax(dim=-1) # la mejor de las 3 máscaras de cada caja
    silhouette = masks[torch.arange(len(boxes)), best].any(dim=0).numpy()

alpha = (silhouette * 255).astype(np.uint8)
Image.fromarray(np.dstack([np.array(image), alpha])).save("sin_fondo.png")
print(len(boxes), "persona(s), silueta de", int(silhouette.sum()), "píxeles")
```

```
# Profundidad: distancia relativa
from transformers import pipeline

depth = pipeline("depth-estimation", model="depth-anything/Depth-Anything-V2-Small-hf")
result = depth("mi_imagen.jpg")
result["depth"].save("profundidad.png") # claro es cerca, oscuro lejos
```

```
# Matching: correspondencias entre dos fotos
from transformers import pipeline

matcher = pipeline("keypoint-matching", model="ETH-CVG/lightglue_superpoint")
matches = matcher(["vista_1.jpg", "vista_2.jpg"], threshold=0.9)
print(len(matches), matches[0]) # un punto en cada foto y su puntuación
```

- Cada tarea es independiente: ejecuta solo la que necesites y no cargues el resto.
- Revisa la licencia antes de llevarlo a producción. **LightGlue con SuperPoint no permite uso comercial** sin acuerdo; el resto de recetas usa modelos con licencia Apache 2.0.
- Para encadenar dos, empieza por profundidad y usa el mapa como máscara: es la combinación más directa.

Las recetas están ejecutadas el 23 de octubre de 2026 con transformers 5.20, la versión que trae Colab. En la versión 5 ya no existen las tareas de pipeline image-to-text ni image-to-image, y nunca ha existido una de keypoint-detection: por eso pose, OCR y superresolución cargan el modelo directamente, igual que el cuaderno. Para quitar el fondo el curso usaba RMBG-1.4, que ejecuta código de su repositorio y no permite uso comercial; ahora se encadenan RT-DETR y SAM 2. El cuaderno ejecuta cinco de las seis tareas con imágenes de ejemplo, todas menos matching, y deja el encadenado como ejercicio.

Antes de usar cualquiera de estas seis

Seis comprobaciones comunes a tareas muy distintas.

☐ **Sabes qué es medida y qué es generación**

Pose y profundidad estiman; superresolución sintetiza. No se usan igual como evidencia.

☐ **La licencia permite tu uso**

Varias de estas familias tienen restricciones comerciales. Compruébalo por modelo.

☐ **La escala está resuelta**

Si necesitas metros, un mapa relativo no sirve: hace falta calibración o un modelo métrico.

☐ **Probado con tus imágenes**

Resolución, encuadre y calidad reales, no las fotos de la demo.

☐ **Sin diagnóstico automático**

En medicina o seguridad, revisión humana sobre la imagen original.

☐ **Coste por imagen calculado**

Si vas a procesar miles, mide tiempo y memoria antes de comprometerte.

Cinco tareas, un cuaderno

Pose, OCR, superresolución, fondo y profundidad sobre tus propias imágenes.

1. Guarda una copia en Drive.
2. Ejecuta cada tarea por separado.
3. Combina dos pipelines y comenta el resultado.

[Abrir en Colab ↗](#)

Cada tarea es independiente: puedes ejecutar solo la que te interese. La ampliación consiste en encadenar dos pipelines, por ejemplo estimar profundidad y usar un umbral para recortar el fondo, o pasar un recorte con superresolución al OCR. Anota qué modelo, qué versión y qué error observas en cada paso.