OpenCV - Visión por Computadora y Machine Learning

Freddy Rospigliosi Cohaila

Agenda

- OpenCV
- Filtros y bordes (Canny)
- Detección de objetos (Haar Cascade Classifier)
- Introducción Machine Learning en dispositivos de baja potencia (Arduino, Raspberry Pi, etc).

¿Qué es OpenCV?

- Librería de código abierto para visión por computadora
  - Compatible con Python, C++, Java
  - Librería muy usada para:
    - Detección de rostros
    - Seguimiento de movimiento
    - Reconocimiento de objetos
    - Procesamiento de imágenes en tiempo real

- OpenCV = *Open Source Computer Vision*

Visión por computadora

- La visión por computadora (también conocida como visión artificial o visión por ordenador) es un campo de la inteligencia artificial que se enfoca en enseñar a las computadoras a "ver", interpretar y comprender el mundo visual de la misma manera que lo hacen los humanos. En esencia, se trata de permitir que las máquinas adquieran, procesen, analicen y extraigan información significativa de imágenes y videos.

¿Cómo funciona?

- Las computadoras "ven" a través de cámaras que capturan imágenes o videos. Estas imágenes se convierten en datos digitales (generalmente en formato de píxeles, como RGB) que luego son procesados y analizados. Para que una computadora pueda interpretar lo que "ve", se utilizan diversas técnicas, incluyendo:
- **Procesamiento de imágenes:** Mejora la calidad de la imagen, elimina el ruido, ajusta el contraste, etc.
- **Segmentación:** Divide la imagen en regiones o "objetos" de interés.
- **Extracción de características:** Identifica elementos relevantes como bordes, texturas, formas y patrones.
- **Reconocimiento y clasificación:** Asigna etiquetas o categorías a los objetos identificados, basándose en patrones aprendidos de miles de imágenes previas. Esto a menudo se logra mediante algoritmos de aprendizaje automático y, en particular, redes neuronales convolucionales (CNN).
- **Seguimiento de objetos:** Sigue el movimiento de objetos a lo largo del tiempo en una secuencia de imágenes (video).

Objetivos clave

- **Automatizar tareas**: Permitir que las máquinas realicen tareas que tradicionalmente requerían la visión humana.
- **Extraer información**: Obtener datos útiles y comprensibles de imágenes y videos.
- **Tomar decisiones**: Permitir que los sistemas actúen o tomen decisiones basadas en la información visual que han procesado.

Aplicaciones de la visión por computadora

- La visión por computadora tiene una amplia gama de aplicaciones en diversos campos:
- **Industria y manufactura:**
  - **Control de calidad:** Detección de defectos en productos.
  - **Automatización de procesos:** Guiado de robots para tareas de ensamblaje o manipulación.
  - **Inspección visual automatizada:** Revisar miles de piezas por minuto con alta precisión.
- **Seguridad y vigilancia:**
  - **Detección de intrusos**: Identificación de personas o vehículos en zonas restringidas.
  - **Reconocimiento facial y biométrico**: Para control de acceso y verificación de identidad.
  - **Análisis de video**: Monitoreo y detección de patrones de comportamiento sospechosos.

Aplicaciones de la visión por computadora

- **Automoción**: Vehículos autónomos: Identificación de otros autos, peatones, señales de tráfico, marcas de carril y obstáculos para una conducción segura.
- **Medicina y salud: Diagnóstico médico:** Detección de lesiones o enfermedades en imágenes médicas (rayos X, resonancias magnéticas).
- **Asistencia en cirugías:** Guiado de instrumentos quirúrgicos.
- **Agricultura y alimentación**: Detección de enfermedades y plagas en cultivos.
- **Cosecha y clasificación de productos**: Basado en madurez, tamaño, color y forma.
- **Robótica**: Permite a los robots "ver" y navegar en su entorno, interactuar con objetos y realizar tareas complejas.
- **Realidad aumentada y virtual**: Integra elementos digitales con el mundo real, utilizando la visión por computadora para una mezcla fluida.

Aplicaciones cotidianas:

- Filtros en fotos y videos (ej. filtros de Snapchat o Instagram).
- Traducción en tiempo real de texto en imágenes (Google Translate).
- Organización y búsqueda de fotos por contenido.

¿Qué es Canny?

- Algoritmo de detección de bordes
- El algoritmo de detección de bordes de Canny, desarrollado por John F. Canny en 1986, es una de las técnicas más influyentes y ampliamente utilizadas en el campo del procesamiento de imágenes y la visión por computadora. Su objetivo es identificar los bordes significativos en una imagen de manera precisa y robusta, minimizando el impacto del ruido.

- Piensa en los bordes como los lugares en una imagen donde hay un cambio abrupto en la intensidad de los píxeles. Estos cambios suelen corresponder a los límites de los objetos, las líneas o las texturas. Canny no es un algoritmo de un solo paso; es un proceso de múltiples etapas que combina varios pasos de procesamiento para lograr resultados óptimos.

Etapas del algoritmo

- **Suavizado de la imagen (Reducción de Ruido):**
- Los bordes son muy sensibles al ruido en una imagen (variaciones aleatorias en la intensidad de los píxeles). Para evitar que el ruido se detecte como un borde falso, el primer paso es aplicar un filtro de suavizado.

Cálculo del Gradiente de Intensidad:

- Una vez que la imagen está suavizada, el siguiente paso es encontrar la "fuerza" y "dirección" de los cambios de intensidad en cada píxel. Esto se hace calculando el gradiente de la imagen.

Supresión de No-Máximos (Non-Maximum Suppression - NMS):

- El cálculo del gradiente a menudo produce bordes "gruesos", donde varios píxeles adyacentes a lo largo de un borde tienen una alta magnitud de gradiente. Canny busca obtener bordes de un solo píxel de grosor.
- La supresión de no-máximos examina la magnitud del gradiente de cada píxel en la dirección de su gradiente. Si el píxel actual no es un máximo local en esa dirección, se suprime (se establece a cero), es decir, no se considera un borde.

Umbralización por Histéresis (Hysteresis Thresholding)

- Después de la supresión de no-máximos, todavía podemos tener algunos bordes "falsos" o ruido residual, junto con bordes verdaderos que podrían ser un poco débiles.

Ventajas del algoritmo Canny

- **Buena detección**: Detecta la mayoría de los bordes reales en la imagen.
- **Buena localización**: Los bordes detectados están cerca de los bordes reales.
- **Respuesta única**: Un solo borde del objeto solo debería producir una única respuesta de borde.
- **Minimización de falsos positivos**: Es robusto al ruido gracias al suavizado Gaussiano y la umbralización por histéresis.
- **Bordes delgados**: Produce bordes de un píxel de ancho.

Aplicaciones de Canny

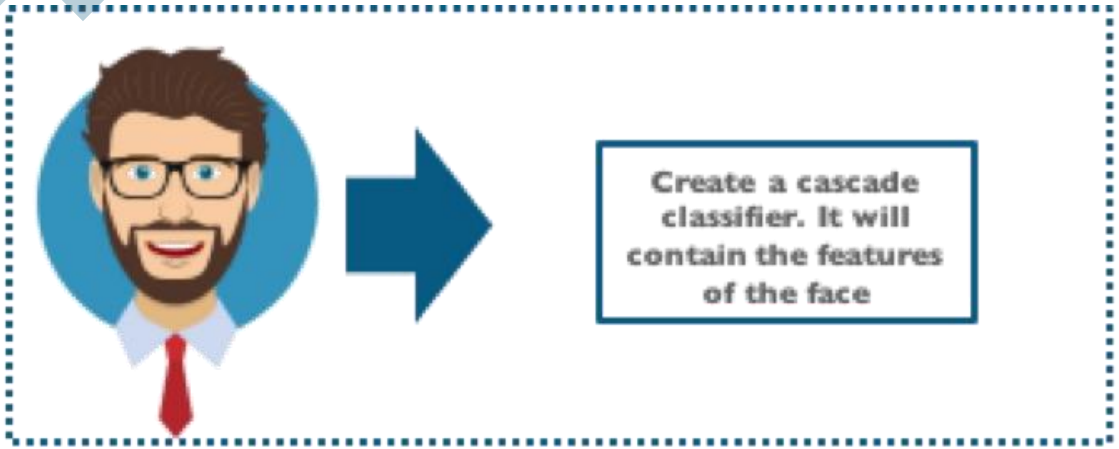
- **Reconocimiento de objetos**: Ayuda a aislar las formas de los objetos.
- **Segmentación de imágenes**: Delimita las regiones de interés.
- **Detección de características**: Proporciona un punto de partida para encontrar puntos clave en una imagen.
- **Visión robótica y navegación**: Los robots pueden usar los bordes para entender su entorno y evitar obstáculos.
- **Sistemas de asistencia al conductor**: Para la detección de carriles y objetos.

Cascade Classifier – Clasificador de cascada Haar

- Es un método muy popular y eficiente en visión por computadora para la **detección de objetos** en imágenes y videos. Fue propuesto por Paul Viola y Michael Jones en 2001 en su influyente paper "Rapid Object Detection using a Boosted Cascade of Simple Features".
- Este clasificador es particularmente conocido por su uso en la **detección facial** en tiempo real, aunque se puede entrenar para detectar cualquier tipo de objeto. Su principal fortaleza es su **velocidad** y su capacidad para funcionar en tiempo real, incluso en sistemas con recursos limitados.

Adaboost (Aprendizaje Potenciado)

- Una sola característica tipo Haar es un "clasificador débil", es decir, no es muy bueno para clasificar si una región contiene un objeto o no. Sin embargo, el algoritmo **AdaBoost** se utiliza para combinar miles de estos clasificadores débiles en un solo "clasificador fuerte" y robusto.
- **AdaBoost** selecciona las características tipo Haar más relevantes y les asigna pesos. Aquellas características que mejor distinguen entre objetos (por ejemplo, caras) y no-objetos reciben pesos más altos. De esta manera, se construye un clasificador que, en conjunto, es muy preciso.



Create a cascade classifier. It will contain the features of the face

OpenCV

Input Image

Displayed Image

```
[[[121,131,213]
[213,143,176]
[125,127,128]
.....
.....
[134,135,172]
[213,143,176]
[213,143,176]]]
```

OpenCV will read the input image and the feature file

El proceso funciona así:

- Una ventana de detección (un subconjunto de la imagen) se desliza por toda la imagen.
- Cada ventana pasa por la primera etapa del clasificador en cascada.
- Si la ventana no pasa la primera etapa (es decir, el clasificador de la primera etapa determina que no es el objeto de interés), se descarta inmediatamente.
- Si la ventana pasa la primera etapa, se pasa a la segunda etapa, y así sucesivamente.
- Solo las ventanas que pasan **todas las etapas** de la cascada se clasifican como que contienen el objeto de interés (por ejemplo, una cara).