**📘 Práctica 1 — Análisis de Imágenes en Python**

**Autores: *Portocarrero Rodriguez Habid***

**Equipo: Escuadrón 241  
Materia: Visión por Computadora  
Instituto / Profesor: *Maria Cruz Elena*  
Fecha: *22.09.2025*  
Entorno: Python 3.9.18 (Anaconda, env Nirvana) en Windows**

**1) Resumen Ejecutivo**

**Este proyecto carga una imagen de ejemplo (imagen1.jpg, una rosa), la convierte a arreglos NumPy y aplica transformaciones básicas de visión: visualización RGB, conversión a escala de grises, conversión a HSV, cálculo de histogramas y binarización por umbral configurable (por defecto 128). Los resultados se muestran con Matplotlib y sirven como base para prácticas posteriores (segmentación y mejoras de contraste). Se documenta el entorno usado (Anaconda/Nirvana) y la solución a errores típicos de dependencias.**

**2) Objetivo(s) y Alcance**

* **Objetivo general: Implementar y documentar un flujo mínimo de análisis de imágenes en Python para cargar, transformar y segmentar una imagen mediante umbralización.**
* **Objetivos específicos:**
  + **Cargar imágenes con Pillow y convertirlas a numpy.ndarray.**
  + **Visualizar canales y resultados con Matplotlib.**
  + **Implementar conversión RGB→Grayscale y RGB→HSV.**
  + **Aplicar umbral fijo (parámetro umbral) y discutir alternativas (Otsu).**
  + **Dejar instrucciones reproducibles del entorno y dependencias.**
* **Alcance: Se trabaja con una imagen local (imagen1.jpg). No incluye entrenamiento de modelos ni procesamiento en lote.**

**3) Contexto y Motivación**

* **Problema a resolver (por qué es relevante).**
* **Casos de uso (laboratorio, seguridad, industria, etc.).**

**4) Dataset y Fuentes de Datos**

* **Fuente: archivo local imagen1.jpg ubicado en la carpeta Practica1/.**
* **Formato: JPEG (960×1280 aprox.).**
* **Notas: Para reproducir, colocar cualquier imagen JPG con ese nombre en la misma carpeta del script o ajustar la ruta en el código (ver sección Ejecución/ Uso).**

**5) Metodología / Pipeline**

1. **Carga: PIL.Image.open(path).convert("RGB") → numpy.array.**
2. **Visualización inicial: mostrar RGB y canales individuales.**
3. **Conversión a Grayscale: luminancia BT.601 con OpenCV (cv2.cvtColor).**
4. **Binarización: umbral fijo (cv2.threshold).**
5. **Modelos de color: YIQ, CMY y HSV con skimage.color.**
6. **Histogramas interactivos: cada figura incluye un botón "Histogramas" que abre una ventana única con los histogramas de los ítems relevantes de esa figura.**
7. **Salida: figuras y ventanas de histogramas (bloqueantes) para cada etapa.**

**Diagrama simple**

**imagen1.jpg → Carga (Pillow) → Arrays (NumPy) → RGB/Gris → Umbral → {YIQ, CMY, HSV} → Visualización + Botón de histogramas**

**6) Arquitectura del Proyecto (Estructura de Carpetas)**

**vision-proyecto/**

**├─ src/**

**| ├─ análisis\_imagenes.py**

**| ├─ Documentacio\_practica1\_escuadron241.docx**

**| ├─imagen1.jpg**

**7) Instalación y Requisitos**

**Hardware: PC con Windows.  
Software: VS Code + Anaconda. Entorno: Nirvana (Python 3.9.18).**

**Paquetes clave y versiones (probadas):**

**numpy==1.26.4**

**matplotlib==3.7.3**

**pillow==11.1.0**

**# opcionales útiles**

**opencv-python==4.9.0.80**

**scikit-image==0.21.0**

**scipy==1.10.1**

**Instalación (usando el Python del entorno Nirvana):**

**"C:\Users\Maria Guadalupe\anaconda3\envs\Nirvana\python.exe" -m pip install --no-cache-dir numpy==1.26.4 matplotlib==3.7.3 pillow**

**# opcional**

**"C:\Users\Maria Guadalupe\anaconda3\envs\Nirvana\python.exe" -m pip install --no-cache-dir opencv-python==4.9.0.80 scikit-image==0.21.0 scipy==1.10.1**

**8) Configuración**

**configs/default.yaml (ejemplo)**

**input:**

**source: "data/samples/video.mp4" # ruta de video o carpeta de imágenes**

**img\_size: 640**

**model:**

**weights: "models/best.pt"**

**conf\_thres: 0.25**

**iou\_thres: 0.45**

**tracking:**

**enabled: true**

**max\_age: 30**

**min\_hits: 3**

**output:**

**save\_video: true**

**save\_csv: true**

**out\_dir: "runs/exp1"**

**9) Ejecución / Uso**

**Opción A (recomendada): ejecutar desde la carpeta Practica1/:**

**cd "C:\Users\Maria Guadalupe\Desktop\GitHub\vision-proyecto\Practica1"**

**"C:\Users\Maria Guadalupe\anaconda3\envs\Nirvana\python.exe" .nalisis\_imagenes.py**

**Opción B: pasar ruta absoluta en el código (ejemplo):**

**demo(r"C:\Users\Maria Guadalupe\Desktop\GitHub\vision-proyecto\Practica1\imagen1.jpg", umbral=128)**

**(Opcional) Llamada robusta dentro del script:**

**import os**

**BASE = os.path.dirname(os.path.abspath(\_\_file\_\_))**

**path\_img = os.path.join(BASE, "imagen1.jpg")**

**demo(path\_img, umbral=128)**

**10) API Pública (Funciones/Clases Clave)**

* **to\_uint8(img): normaliza cualquier imagen a uint8 0–255 sin alterar su apariencia visual.**
* **plot\_hist\_on\_axes(ax, img\_u8, name): dibuja histograma; una curva para gris, tres para RGB.**
* **open\_hist\_grid(images\_dict, title, hist\_whitelist): abre una sola ventana con un grid de histogramas para claves seleccionadas.**
* **figure\_with\_hist\_button(images\_dict, title, layout, cmaps, hist\_whitelist): crea figura con imágenes y un botón "Histogramas" que invoca open\_hist\_grid.**
* **cargar\_rgb(path): carga con Pillow en RGB uint8.**
* **separar\_canales(rgb): devuelve matrices R, G, B.**
* **a\_grises\_bt601(rgb): conversión a gris por BT.601 usando OpenCV.**
* **binarizar(gray\_u8, t): umbral fijo con OpenCV.**
* **convertir\_yiq(rgb\_u8): usa skimage para obtener Y, I, Q y un compuesto visual.**
* **convertir\_cmy(rgb\_u8): modelo sustractivo simple CMY (255–R/G/B).**
* **convertir\_hsv(rgb\_u8): HSV con skimage.color y reconstrucción HSV→RGB para vista.**
* **demo(path, umbral): orquesta el flujo completo en 7 pasos con figuras consecutivas y sus histogramas.**

**11) Evaluación y Métricas**

* **Clasificación: accuracy, precision, recall, F1.**
* **Detección: mAP@[.5:.95], IoU, FPS.**
* **Tracking: MOTA, MOTP, IDF1, ID switches.**
* **Curvas: PR, ROC, confusión.**

**Incluye tablas y gráficas (pega imágenes o enlaza a runs/).**

**12) Experimentos**

* **Hipótesis y variables (ablation): *sin* sustracción de fondo vs *con*, img\_size 640 vs 960, etc.**
* **Configuraciones probadas (tabla).**
* **Resultados por experimento (tabla).**
* **Discusión breve.**

**13) Resultados / Visualizaciones**

* **Capturas de pantalla (detecciones, trayectorias).**
* **Videos anotados.**
* **Ejemplos de fallos (falsos positivos/negativos) y explicación.**

**14) Rendimiento y Optimización**

* **FPS promedio (CPU/GPU), uso de memoria.**
* **Técnicas: batch inferencing, half precision, ONNX/TensorRT, multithreading.**
* **Perfilado (line-profiler, cProfile) y hallazgos.**

**15) Pruebas (Testing)**

* **Unitarias: pytest (I/O, preprocesado, parsers).**
* **Integración: pipeline end-to-end con un video corto.**
* **Determinismo: fijar semillas aleatorias.**

**tests/test\_io.py (ejemplo)**

**import pytest**

**from src.utils.io import load\_video**

**def test\_load\_video\_not\_found():**

**with pytest.raises(FileNotFoundError):**

**load\_video("no\_existe.mp4")**

**16) Registro y Manejo de Errores**

* **Logging con niveles (INFO/DEBUG/ERROR).**
* **Mensajes claros para archivos faltantes, cámaras no disponibles, pesos incompatibles.**
* **Mecanismos de retry y tiempo de espera.**

**logging (snippet)**

**from loguru import logger**

**logger.add("runs/app.log", rotation="10 MB")**

**logger.info("Inicio de la aplicación")**

**17) Limitaciones y Trabajo Futuro**

* **Condiciones donde falla (iluminación, oclusiones, velocidad).**
* **Mejoras planeadas (más datos, finetuning, nuevos algoritmos, mejores métricas).**

**18) Consideraciones Éticas y de Privacidad**

* **Tratamiento de datos personales en video.**
* **Almacenamiento seguro y retención.**
* **Minimización de datos y anonimización (blur de rostros, p.ej.).**

**19) Cómo Reproducir**

1. **Clonar el repo o copiar la carpeta del proyecto.**
2. **En VS Code, seleccionar intérprete: anaconda3/envs/Nirvana/python.exe.**
3. **Instalar dependencias probadas (ver Instalación y Requisitos).**
4. **Colocar imagen1.jpg en Practica1/ (o ajustar la ruta en el script).**
5. **Ejecutar el comando de la sección Ejecución / Uso.**

**20) FAQ / Troubleshooting**

* **ModuleNotFoundError: No module named 'PIL' → instalar Pillow en el entorno activo: python -m pip install pillow.**
* **Choque NumPy 2.x vs Matplotlib ("\_ARRAY\_API not found" / "multiarray failed to import") → usar numpy==1.26.4 y matplotlib==3.7.3.**
* **ImportError de backend Qt5Agg → instala PyQt5 o cambia a TkAgg en la línea de backend.**
* **FileNotFoundError: 'imagen1.jpg' → ejecutar desde Practica1 o usar ruta absoluta / solución robusta con os.path.**
* **Se abren muchas ventanas de histogramas → el botón abre una sola ventana por figura; ciérrala antes de pasar a la siguiente.**

**21) Licencia y Créditos**

* **Licencia del código (MIT/GPL/…)**
* **Créditos a datasets/librerías.**

**22) Anexos**

* **Fórmulas (IoU, mAP, MOTA).**
* **Diagramas adicionales.**
* **Tablas extensas de resultados.**

**(Opcional) Documentación Autogenerada**

* **Docstrings + Sphinx/MkDocs:**

**# Sphinx**

**pip install sphinx sphinx-rtd-theme**

**sphinx-quickstart docs**

**# Configura theme y autodoc; luego:**

**make html**

**# MkDocs**

**pip install mkdocs mkdocs-material**

**mkdocs new .**

**mkdocs serve**

**(Opcional) Calidad de Código**

**pip install black isort flake8 pre-commit**

**pre-commit install**