

PhotoLab Express

Una aplicación web revolucionaria para el procesamiento de imágenes por lotes, construida con una interfaz gráfica interactiva usando Streamlit. PhotoLab Express permite a los usuarios aplicar una variedad de filtros avanzados, detectar rostros y clasificar imágenes utilizando inteligencia artificial de última generación. El sistema está diseñado para ser eficiente, utilizando procesamiento en paralelo, caché inteligente y optimizaciones de rendimiento que aceleran significativamente las operaciones.

Arquitectura del Proyecto

PhotoLab Express está organizado en una estructura modular cuidadosamente diseñada para facilitar el mantenimiento, la escalabilidad y la colaboración en equipo. El proyecto se divide en componentes especializados, cada uno responsable de una función específica dentro del sistema.

app.py

Punto de entrada principal. Contiene la lógica de la interfaz gráfica de usuario construida con Streamlit, incluyendo cargadores de archivos, menús de selección y visualización de resultados.

src/pipeline.py

Orquesta el procesamiento de imágenes en paralelo, distribuyendo tareas entre múltiples núcleos de CPU para maximizar el rendimiento.

src/filters.py

Contiene funciones para aplicar filtros avanzados a las imágenes, incluyendo Sobel, Canny y desenfoque Gaussiano.

src/detect.py

Implementa la lógica para la detección de rostros utilizando cascadas de clasificadores pre-entrenados.

src/ml.py

Gestiona la clasificación de imágenes con el modelo CLIP de OpenAI, permitiendo etiquetado semántico flexible.

src/cache.py

Implementa el sistema de caché para acelerar el procesamiento y evitar cálculos redundantes.

src/io_utils.py

Utilidades para leer, escribir y manipular imágenes en diferentes formatos.

src/gen.py

Generadores para el procesamiento de imágenes por bloques, optimizando el uso de memoria.

Además, el proyecto incluye directorios para pruebas unitarias (tests/), datos de referencia (data/), análisis de rendimiento (profiling/) y un archivo requirements.txt que define todas las dependencias necesarias.

Concurrencia y Paralelismo

PhotoLab Express implementa procesamiento paralelo de imágenes para aprovechar al máximo los recursos disponibles del sistema. La función `run_pipeline` en `src/pipeline.py` utiliza `concurrent.futures.ProcessPoolExecutor`, que distribuye automáticamente el procesamiento de imágenes entre todos los núcleos de CPU disponibles mediante `os.cpu_count()`.

Esta estrategia es ideal para tareas intensivas en CPU como el procesamiento de imágenes, ya que evita el Bloqueo Global del Intérprete (GIL) de Python. El GIL es una limitación fundamental de Python que impide que múltiples threads ejecuten código Python simultáneamente. Sin embargo, los procesos son completamente independientes y pueden ejecutarse verdaderamente en paralelo en diferentes núcleos de CPU, lo que resulta en mejoras de rendimiento casi lineales con el número de núcleos disponibles.

Esta implementación permite que PhotoLab Express procese lotes de cientos o incluso miles de imágenes de manera significativamente más rápida que un procesamiento secuencial tradicional.

Procesamiento por Lotes y Clasificación Inteligente

PhotoLab Express está diseñado específicamente para manejar el procesamiento de múltiples imágenes de manera eficiente. La interfaz de usuario permite la selección de múltiples archivos o carpetas completas mediante `st.file_uploader`, facilitando la carga de lotes grandes de imágenes.

La función `run_pipeline` en `src/pipeline.py` está optimizada para recibir una lista completa de rutas de imágenes y procesarlas en un solo lote coordinado. Más importante aún, la función `classify_batch` en `src/ml.py` procesa un lote completo de imágenes en una sola pasada a través del modelo CLIP de OpenAI, lo cual es exponencialmente más eficiente que procesarlas una por una.

1 Carga de Imágenes

Los usuarios seleccionan múltiples archivos o carpetas completas a través de la interfaz intuitiva de Streamlit.

2 Aplicación de Filtros

Se aplican los filtros seleccionados a todas las imágenes del lote de manera paralela.

3 Detección de Rostros

Si está habilitada, se detectan y marcan rostros en todas las imágenes del lote.

4 Clasificación Semántica

CLIP clasifica todas las imágenes y genera etiquetas para organización automática en carpetas.

Generadores y Optimización de Memoria

Para manejar imágenes extremadamente grandes sin consumir cantidades excesivas de memoria RAM, PhotoLab Express implementa generadores en `src/gen.py`. La función `tiles` es un generador que divide una imagen en bloques más pequeños, permitiendo el procesamiento incremental de imágenes masivas.

Los generadores son una característica poderosa de Python que permite iterar sobre datos sin cargar todo en memoria simultáneamente. Esto es especialmente valioso en aplicaciones de procesamiento de imágenes donde las imágenes pueden tener dimensiones de miles de píxeles. En lugar de cargar toda la imagen en memoria, el generador produce bloques bajo demanda, procesa cada bloque y descarta los datos innecesarios.

Aunque el generador de bloques no está actualmente integrado en la pipeline principal, está disponible para ser utilizado en filtros que requieran procesamiento por bloques. Esta arquitectura flexible permite que PhotoLab Express escale a imágenes de resolución arbitraria sin limitaciones de memoria, manteniendo un rendimiento consistente incluso en sistemas con recursos limitados.

Sistema de Caché Inteligente

PhotoLab Express implementa un sistema de caché sofisticado utilizando la librería `diskcache` para evitar el reprocesamiento innecesario de imágenes. El decorador `@memoize` en `src/cache.py` se aplica a funciones de procesamiento intensivo en `src/filters.py` y `src/detect.py`.

Cuando una imagen se procesa con parámetros específicos, el resultado se guarda automáticamente en el disco. Si la misma imagen se procesa nuevamente con los mismos parámetros, el resultado se recupera instantáneamente de la caché en lugar de ser recalculado. Esta estrategia reduce drásticamente el tiempo de ejecución, especialmente cuando se trabaja con lotes grandes o cuando se experimentan diferentes combinaciones de filtros.

Librerías Especializadas:

PhotoLab Express utiliza una combinación cuidadosamente seleccionada de librerías especializadas, cada una optimizada para su función específica. **Streamlit** proporciona la interfaz web interactiva con muy pocas líneas de código, eliminando la necesidad de escribir HTML, CSS o JavaScript. **OpenCV**, escrito en C++ y optimizado para rendimiento, es el estándar de la industria para visión por computadora, manejando lectura, escritura y procesamiento de imágenes con velocidad incomparable.

NumPy es fundamental para operaciones matemáticas en arrays de píxeles, con implementaciones en C que son órdenes de magnitud más rápidas que Python puro. **Pillow (PIL)** actúa como puente entre formatos de imagen, convirtiendo entre OpenCV y PIL para compatibilidad con Streamlit y CLIP. **Transformers, Torch y Torchvision** forman el núcleo de la funcionalidad de IA, permitiendo usar el modelo CLIP de OpenAI con solo unas pocas líneas de código, aprovechando aceleración por GPU cuando está disponible.



Streamlit

Interfaz web interactiva



OpenCV

Procesamiento de imágenes



NumPy

Operaciones matemáticas



Transformers

Modelos de IA avanzados

Pandas organiza resultados en DataFrames tabulares para visualización y exportación CSV. **DiskCache** implementa persistencia de caché en disco. **Pytest** automatiza pruebas unitarias con imágenes de referencia ("golden images") para detectar regresiones. **Scikit-learn** proporciona algoritmos de machine learning complementarios para pre-procesamiento avanzado.

Inteligencia Artificial con CLIP de OpenAI

PhotoLab Express integra el modelo **CLIP** (Contrastive Language-Image Pre-training) de OpenAI, una red neuronal revolucionaria que entiende tanto imágenes como texto. A diferencia de los modelos de clasificación tradicionales limitados a categorías predefinidas, CLIP puede clasificar imágenes basándose en etiquetas de texto flexibles y contextuales, permitiendo una clasificación mucho más precisa y adaptable.

La función `classify_batch` en `src/ml.py` procesa lotes completos de imágenes a través de CLIP en una sola pasada, generando etiquetas semánticas que se utilizan automáticamente para organizar las imágenes procesadas en carpetas temáticas. El sistema puede reconocer y categorizar imágenes sin necesidad de entrenamiento previo en esas categorías específicas.

La integración de CLIP transforma PhotoLab Express de una simple herramienta de procesamiento de imágenes en un sistema inteligente capaz de entender el contenido semántico de las imágenes. Esto permite una organización automática y contextual de los resultados, mejorando significativamente la experiencia del usuario y la utilidad de la aplicación para flujos de trabajo profesionales.

Pruebas, Profiling y Garantía de Calidad

PhotoLab Express implementa un enfoque riguroso de aseguramiento de calidad mediante pruebas automatizadas y análisis de rendimiento. El directorio `tests/` contiene pruebas unitarias usando `pytest`, con un enfoque innovador de "imágenes de referencia" (golden images). En la primera ejecución, se generan imágenes de referencia. En ejecuciones posteriores, los resultados de los filtros se comparan automáticamente con estas imágenes para detectar cualquier regresión o cambio inesperado.

El directorio `profiling/` contiene scripts para análisis detallado de rendimiento. El script `run_profile.sh` ejecuta `cProfile` en la aplicación, generando datos de profiling que pueden visualizarse con herramientas como `snakeviz` para análisis interactivo. Esto permite identificar cuellos de botella de rendimiento y optimizar las secciones críticas del código.

01

Pruebas Unitarias

Pytest valida cada componente de forma aislada con casos de prueba exhaustivos

02

Imágenes de Referencia

Golden images detectan regresiones visuales en filtros y procesamiento

03

Profiling de Rendimiento

`cProfile` identifica cuellos de botella y oportunidades de optimización

04

Análisis Interactivo

`snakeviz` visualiza datos de profiling para comprensión profunda del rendimiento

Pruebas

Objetivo: asegurar que los filtros, la detección de rostros y las funciones del pipeline funcionen correctamente y no generen resultados inesperados.

test_filters.py: valida todos los filtros de imagen usando golden images para detectar cualquier cambio inesperado en la salida.

test_detect.py: convierte la imagen en escalas de grises para comprobar la detección de rostros y el cálculo de colores dominantes funcionen correctamente.

Se agregaron validaciones de forma, tipo de datos y consistencia para garantizar resultados fiables.



Conclusión

PhotoLab Express representa una aplicación de procesamiento de imágenes, construida con principios de ingeniería de software avanzada. La combinación de procesamiento paralelo, caché inteligente, generadores eficientes en memoria y librerías especializadas crea un sistema capaz de manejar cargas de trabajo masivas con rendimiento excepcional.

La arquitectura modular permite mantenimiento sencillo y escalabilidad futura. La integración de CLIP de OpenAI proporciona capacidades de inteligencia artificial de última generación para clasificación semántica automática. Las pruebas rigurosas y el profiling de rendimiento garantizan confiabilidad y eficiencia continua. PhotoLab Express demuestra cómo las mejores prácticas de ingeniería de software, combinadas con tecnologías modernas, pueden crear aplicaciones que son simultáneamente poderosas, eficientes y fáciles de usar.



Rendimiento

Procesamiento paralelo en múltiples núcleos de CPU para velocidad máxima



Eficiencia

Caché inteligente y generadores para optimización de recursos



Inteligencia

Clasificación semántica con CLIP para organización automática



Calidad

Pruebas exhaustivas y profiling para confiabilidad garantizada

PhotoLab Express

Elige las imágenes

Drag and drop files here
Limit 200MB per file

Browse files

Selfie_2 (2).webp 312.3KB
Selfie_1.webp 450.0KB
Selfie_1 (5).webp 1.1MB

Showing page 1 of 2

1111 House Juri Troy Architects Interior Photography Kitchen Windows Table Countertop Chair .webp
1111 House Juri Troy Architects Interior Photography Stairs .webp
1111 House Juri Troy Architects Interior Photography Windows Beam .webp
Selfie_1 (5).webp
Selfie_1.webp
Selfie_2 (2).webp

Selección de los filtros:

Sobel

Imágenes Procesadas

Imágenes con Rostros Detectados



Imágenes sin Rostros Detectados



Nota: La clasificación por IA se basa en el modelo CLIP de OpenAI. Las etiquetas son más precisas pero aún pueden producir resultados inesperados.

filename	classification
0 1111 House Juri Troy Architects Interior Photography Kitchen Windows Table Countertop Chair .webp	other
1 1111 House Juri Troy Architects Interior Photography Stairs .webp	a photo of a building
2 1111 House Juri Troy Architects Interior Photography Windows Beam .webp	a photo of a building
3 Selfie_1 (5).webp	a photo of a person
4 Selfie_1.webp	a photo of a person
5 Selfie_2 (2).webp	a photo of a person

¡Procesamiento completo!

Diagnóstico de Rendimiento

Gráfico generado por l'instrument para verificar cuellos de botella.



A	B	C
filename		classification
0 1111 House Juri Troy Architects Interior Photography Kitchen Windows Table Countertop Chair .webp		other
1 1111 House Juri Troy Architects Interior Photography Stairs .webp		a photo of a building
2 1111 House Juri Troy Architects Interior Photography Windows Beam .webp		a photo of a building
3 Selfie_1 (5).webp		a photo of a person
4 Selfie_1.webp		a photo of a person
5 Selfie_2 (2).webp		a photo of a person

project.src.filters

Una colección de filtros de procesamiento de imágenes.

Este módulo proporciona un conjunto de funciones para aplicar diversos filtros de procesamiento de imágenes a las imágenes, incluida la detección de bordes, el desenfoque, la nitidez y la manipulación del color. Las funciones están diseñadas para ser utilizadas como parte del pipeline de procesamiento de imágenes.

▶ View Source

def apply_sobel(*args, **kwargs):

▶ View Source

La función envoltorio que implementa la lógica de caché.

def apply_canny(*args, **kwargs):

▶ View Source

La función envoltorio que implementa la lógica de caché.

def apply_gaussian_blur(*args, **kwargs):

▶ View Source

La función envoltorio que implementa la lógica de caché.

def apply_sharpen(*args, **kwargs):

▶ View Source

La función envoltorio que implementa la lógica de caché.

def apply_random_hue_shift(image):

▶ View Source

Aplica un cambio de tono aleatorio a una imagen.

Esta función no está memoizada porque no es determinista.

Args: image (numpy.ndarray): La imagen de entrada.

Returns: numpy.ndarray: La imagen con un cambio de tono aleatorio aplicado.

project.src

Search...

API Documentation

face_cascade
detect_faces()
get_dominant_colors()

built with

project.src.detect

Funciones de detección de objetos y análisis de color.

Este módulo proporciona funciones para detectar objetos en imágenes, como rostros, y para analizar la composición de color de las imágenes, como encontrar los colores dominantes.

▶ View Source

face_cascade = <cv2.CascadeClassifier 000002517F461F70>

def detect_faces(*args, **kwargs):

▶ View Source

La función envoltorio que implementa la lógica de caché.

def get_dominant_colors(*args, **kwargs):

▶ View Source

La función envoltorio que implementa la lógica de caché.