UNIVERSIDAD MARIANO GÁLVEZ DE GUATEMALA

CENTRO EL NARANJO, MIXCO

FACULTAD DE INGENIERÍA EN SISTEMAS Y CIENCIAS DE LA COMPUTACIÓN

INTELIGENCIA ARTIFICIAL

DOCENTE ING. CRISTIAN GIOVANNI MARTINEZ RODRIGUEZ

Imagen que contiene Logotipo

Descripción generada automáticamente

PROYECTO FINAL

Nombres: Carné:

Bryan Ernesto Gámez Cipriano No. 9490-19-3938

Rudy Alexander Amado Soto Rosil No. 9490-20-387

Alfredo Geovanni Ramirez Tzunun No. 9490-20-427

Daniel Antonio Hernández Santos No. 9490-20-1190

Helder Ivan Ajcalon Jacobo No. 9490-20-10326

30 de mayo de 2024

Índice

[Introducción 3](#_Toc168074556)

[Objetivos: 3](#_Toc168074557)

[Descripción del Proyecto: 4](#_Toc168074558)

[Tecnologías Utilizadas: 4](#_Toc168074559)

[Uso del Proyecto: 4](#_Toc168074560)

[Ejemplos de Resultados: 5](#_Toc168074561)

[Estructura del Código 6](#_Toc168074562)

[TiendaIA.py 6](#_Toc168074563)

[Tienda.py 8](#_Toc168074564)

[Observaciones 8](#_Toc168074565)

# Introducción

El proyecto de Reconocimiento de Imágenes en Tienda tiene como objetivo proporcionar una solución automatizada para el reconocimiento y clasificación de productos y billetes dentro de una tienda. Mediante el uso de técnicas avanzadas de inteligencia artificial y visión por computadora, este proyecto puede identificar varios tipos de artículos y denominar billetes en tiempo real utilizando una cámara. Este proyecto integra modelos de detección de objetos preentrenados (YOLO) con OpenCV para proporcionar una solución que puede ser implementada en diversos tipos de tiendas. Utilizando Python como lenguaje de programación principal, el sistema es tanto flexible como escalable.

## Objetivos:

* Automatización de la Identificación de Productos: Mejorar la eficiencia y precisión en el reconocimiento de artículos en una tienda, facilitando el proceso de inventario y ventas.
* Gestión de Pagos Automatizada: Implementar un sistema que detecte y sume automáticamente el valor de los billetes insertados, calculando el cambio necesario.
* Interfaz de Usuario Intuitiva: Proveer una interfaz visual que muestre los productos identificados, sus precios y el balance total, mejorando la experiencia del usuario.

# Descripción del Proyecto:

El sistema captura imágenes en tiempo real utilizando una cámara conectada al equipo. Las imágenes son procesadas para identificar productos y billetes utilizando modelos de detección de objetos YOLO, específicamente la versión YOLOv8. Una vez identificados los objetos, el sistema actualiza una lista de compras visual en pantalla, mostrando los artículos reconocidos junto con sus precios. Además, cuando se detectan billetes, el sistema actualiza el saldo disponible y calcula el cambio en caso de pago.

El código está estructurado de manera modular para facilitar su mantenimiento y expansión futura. Cada componente del sistema, desde la captura de imágenes hasta el procesamiento de pagos, está encapsulado en clases y funciones bien definidas.

Para más detalles y acceso al código completo, visita el repositorio en GitHub: <https://github.com/bryan-egc/Reconocimiento-Imagenes-Tienda>.

## Tecnologías Utilizadas:

* OpenCV: Biblioteca de visión por computadora para la captura, procesamiento y análisis de imágenes.
* YOLO (You Only Look Once): Algoritmo de detección de objetos en tiempo real.
* Python: Lenguaje de programación principal del proyecto.
* Ultralytics: Implementación de YOLO en Python para facilitar el uso de modelos de detección de objetos.
* Modelos YOLOv8: Modelos preentrenados utilizados para la detección de objetos y billetes. Enlace a los modelos

## Uso del Proyecto:

* Ejecutar el Proyecto: Ejecuta el script principal main.py para iniciar el reconocimiento de imágenes.
* Interacción: Usa las teclas 'S' o 's' para sumar el saldo de los billetes detectados y las teclas 'P' o 'p' para procesar el pago.
* Visualización: Los resultados se mostrarán en una ventana de OpenCV con las áreas de compra, pago y lista de compras.

## Ejemplos de Resultados:

* Imagen que contiene interior, pequeño, cuarto, montado

  Descripción generada automáticamenteEjemplo 1: Detección de bananas y botella con sus correspondientes porcentajes de confianza.
* Calendario

  Descripción generada automáticamenteEjemplo 2: Detección de mouse y teclado con sus correspondientes porcentajes de confianza.
* Ejemplo 3: Detección de cuchillo, cuchara y tijeras con sus correspondientes porcentajes de confianza.

Imagen que contiene Pizarra

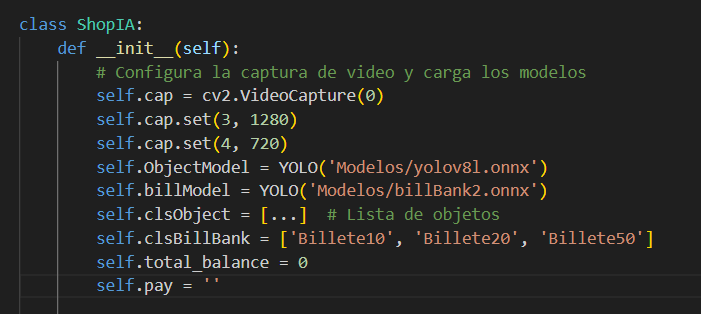
Descripción generada automáticamente

# Estructura del Código

## TiendaIA.py

Este archivo contiene la clase ShopIA, que es el núcleo del sistema de reconocimiento. Incluye:

* Inicialización: Configura la captura de video y carga los modelos de YOLO.
* Funciones de Dibujo: Métodos para dibujar áreas, texto y líneas en las imágenes.
* Gestión de Áreas: Define áreas específicas en la imagen para la detección de objetos y el procesamiento de pagos.
* Lista de Compras: Gestiona los productos detectados y sus precios.
* Procesamiento de Pagos: Calcula el saldo total y el cambio.
* Predicción: Utiliza los modelos de YOLO para detectar objetos y billetes en las imágenes capturadas.
* Función Principal (tiendaIA): Gestiona el flujo de video y coordina la detección y visualización en tiempo real.

Inicialización:

Captura de pantalla de computadora

Descripción generada automáticamenteFunciones de Dibujo:

Texto

Descripción generada automáticamenteGestión de Áreas:

Lista de Compras:

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamenteFunción Principal (tiendaIA):

Para acceder al código completo y ver la estructura de cada módulo, visita el repositorio en GitHub: <https://github.com/bryan-egc/Reconocimiento-Imagenes-Tienda>.

## Tienda.py

Este archivo inicializa y ejecuta la clase ShopIA.

Texto

Descripción generada automáticamente

## Observaciones

* Compatibilidad de Cámara: Asegúrate de que la cámara esté correctamente conectada y funcionando.
* Ubicación de los Modelos: Los modelos deben estar en la carpeta Modelos.
* Requisitos de Hardware: Se recomienda usar una máquina con GPU para un rendimiento óptimo.
* Precisión del Modelo: La precisión puede variar según la calidad del modelo y las imágenes capturadas.
* Entorno de Desarrollo: Se recomienda usar un entorno virtual para manejar las dependencias.
* Actualización de Dependencias: Mantén las dependencias actualizadas para evitar problemas de compatibilidad.
* Errores Comunes: Verifica la correcta instalación de dependencias y ubicación de modelos si surgen problemas.

# Conclusiones

A través de la implementación de modelos de detección de objetos YOLO y el uso de OpenCV para el procesamiento de imágenes, se ha logrado crear un sistema que proporciona una interfaz de usuario intuitiva y visualmente clara. Este proyecto ofrece una valiosa experiencia práctica en el campo de la inteligencia artificial y la visión por computadora, mostrando cómo estas tecnologías pueden ser aplicadas para resolver problemas del mundo real. Los resultados obtenidos validan la viabilidad del sistema para su uso en diversos contextos comerciales, abriendo puertas a futuras mejoras y adaptaciones.