

BGR - Image background remover and object detector

Abstract

This work “Remove and Detect” implements an interactive system for background removal and object detection in images. Background removal is the process of isolating the main object in an image by eliminating everything behind it for further image processing. Objective detection is a computer vision task that involves identifying and locating objects within an image or video. Firstly, it finds the position of the object within an image, represented by a bounding box around the object. Then, it determines what the object is by assigning it a label or category.

The graphical user interface (GUI) is built using Tkinter and CustomTkinter, allowing users to upload images, process them to remove backgrounds, and detect objects. The background removal is handled by rembg, which isolates the foreground and generates an output with a smooth white background. Additionally, the YOLOv8 model is used for object detection, where detected objects are cropped, and the backgrounds of these cropped images are also removed for further processing.

The system provides options to save the processed images and detected objects, offering an intuitive, user-friendly interface to interact with complex image processing techniques. The solution is efficient in handling images with multiple objects and provides a streamlined approach to background removal and object detection through a visually accessible GUI.

Prerequisites

Before running the code, the following libraries and tools are needed to be installed first.

OpenCV	For image processing tasks.
	> pip install opencv-python-headless
NumPy	For array and image manipulation.
	> pip install numpy
Pillow	For handling image input and conversion.
	> pip install pillow
Rembg	For background removal.
	> pip install rembg
Ultralytics	For YOLOv8-based object detection.
	> pip install ultralytics
Torch	Required by Ultralytics for deep learning operations.
	> pip install torch
CustomTkinter	For an improved, modernized GUI interface.
	> pip install customtkinter

The YOLOv8 model will be automatically downloaded when YOLO("yolov8n.pt") is called. An active internet connection is necessary for this step.

System Requirements

- ❖ **Python:** Use Python 3.8 or later.
- ❖ **Operating System:** Works on Windows, Linux, or macOS.
- ❖ **Memory:** Depending on the size of images and the YOLO model, the system should have at least 8GB of RAM.

Setting up directories

There should be 3 folders for the processed images. Those 3 folders should be in the same directory as main.py.

‘result’ folder

The ‘result’ folder is where the background removed image will be stored. There are two images. ‘result.png’ is the background removed image with white background. ‘result_transparent.png’ is the background removed image with transparent background. The ‘result’ folder’s file path can be configured at the variable ‘save_path’ in main.py.

File path example: “C:/Users/HP/PycharmProjects/bgr/result/”

‘cropped’ folder

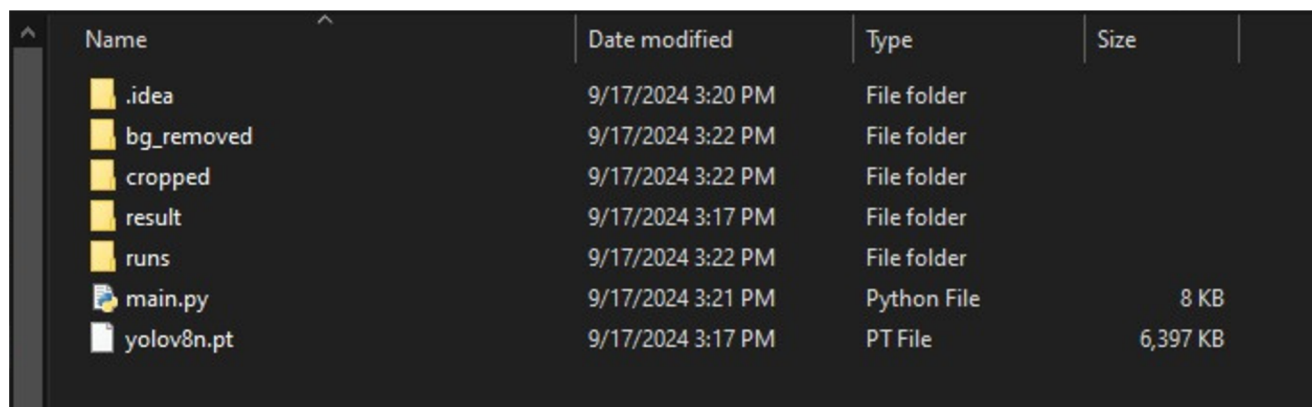
The ‘cropped’ folder is where detected objects in the image are cropped and stored separately with the names of the images being the detected object’s label. The ‘cropped’ folder’s file path can be configured at the variable ‘cropped_path’ in main.py.

File path example: "C:/Users/HP/PycharmProjects/bgr/cropped"

‘bg_removed’ folder

The ‘bg_removed’ folder is where images saved in the ‘cropped’ folder are saved with their backgrounds removed. The ‘bg_removed’ folder’s file path can be configured at the variable ‘bg_removed’ in main.py.

File path example: "C:/Users/HP/PycharmProjects/bgr/bg_removed"

A screenshot of a file explorer window showing the contents of a directory. The table lists files and folders with their names, modification dates, types, and sizes. The folders listed are .idea, bg_removed, cropped, result, and runs. The files listed are main.py (Python File, 8 KB) and yolov8n.pt (PT File, 6,397 KB).

Name	Date modified	Type	Size
.idea	9/17/2024 3:20 PM	File folder	
bg_removed	9/17/2024 3:22 PM	File folder	
cropped	9/17/2024 3:22 PM	File folder	
result	9/17/2024 3:17 PM	File folder	
runs	9/17/2024 3:22 PM	File folder	
main.py	9/17/2024 3:21 PM	Python File	8 KB
yolov8n.pt	9/17/2024 3:17 PM	PT File	6,397 KB

After installing necessary libraries and preparing the file path, now run the code in main.py.

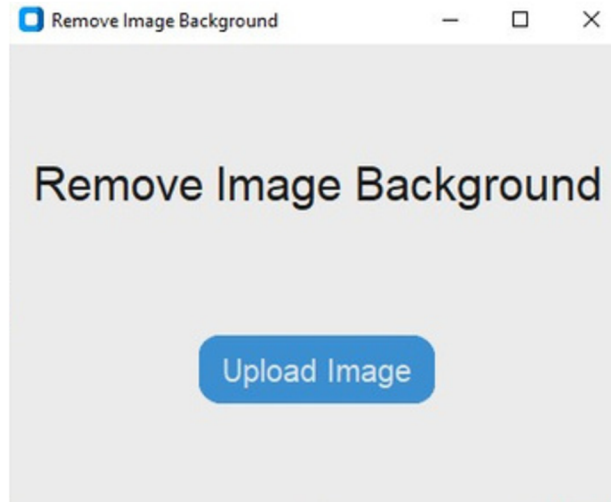


Fig 2. GUI, upload image to be processed

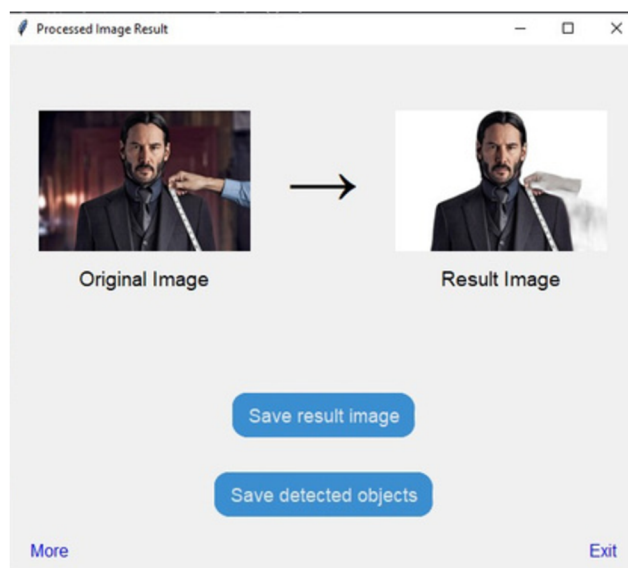


Fig 3. GUI, shows processed image

Steps

- ❖ Choose an image by clicking the “Upload Image” button.
- ❖ The original image and background removed image will be shown as in **Fig 3**.
- ❖ Images can be saved.
 - “**Save result image**” will save the background removed image
 - “**Save detected objects**” will save the objects detected in the image individually
- ❖ To process more images without restarting the program, click “**More**” in the second GUI.
- ❖ To exit the program, click “**Exit**”.

Limitations

- ❖ Can’t clearly detect and remove images with complex foregrounds.
- ❖ Can’t provide the detected image in the GUI yet.

Caution

- ❖ Need to wait a specific duration when running the system.
- ❖ Can overload the CPU as the system is using machine learning models.
- ❖ Evaluation and training data can be opted to show in the terminal when processing the object detection.

```
def save_detected_images(): 1 usage
    if result_img_to_save:
        results = model(save_path2)

        # to train model
        # model.train(data='coco8.yaml', epochs=3)
        # metrics = model.val()
```

Fig 4. Uncomment to train model