GrabCut Image Segmentation Code Explanation

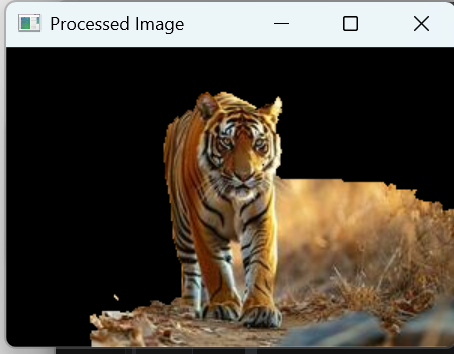
# Foreground Segmentation using GrabCut

import cv2  
import numpy as np  
  
input\_image = cv2.imread(r'C:\Users\Administrator\OneDrive\Desktop\PAI TASKS\image 2.jpg')  
  
if input\_image is None:  
 print("Error: Could not load image. Check the file path.")  
 exit()  
  
segmentation\_mask = np.zeros(input\_image.shape[:2], np.uint8)  
roi = (10, 10, input\_image.shape[1]-10, input\_image.shape[0]-10)  
bg\_model = np.zeros((1, 65), np.float64)  
fg\_model = np.zeros((1, 65), np.float64)  
  
cv2.grabCut(input\_image, segmentation\_mask, roi, bg\_model, fg\_model, 5, cv2.GC\_INIT\_WITH\_RECT)  
  
final\_mask = np.where((segmentation\_mask == 2) | (segmentation\_mask == 0), 0, 1).astype('uint8')  
output\_image = input\_image \* final\_mask[:, :, np.newaxis]  
  
cv2.imshow('Processed Image', output\_image)  
cv2.waitKey(0)  
cv2.destroyAllWindows()  
  
cv2.imwrite(r'C:\Users\Administrator\OneDrive\Desktop\PAI TASKS\processed\_output.jpg', output\_image)  
print("Image saved as processed\_output.jpg")

## Explanation:

This code performs foreground extraction using the GrabCut algorithm in OpenCV.  
  
1. \*\*Import Libraries\*\*: It uses `cv2` (OpenCV) for image processing and `numpy` for array operations.  
2. \*\*Load Image\*\*: Reads the input image from the given file path. If it fails, it prints an error and exits.  
3. \*\*Initialize Mask and Models\*\*:  
 - `segmentation\_mask` is a black mask the same size as the image.  
 - `roi` defines the rectangle area assumed to contain the object.  
 - `bg\_model` and `fg\_model` are used internally by GrabCut.  
4. \*\*GrabCut Algorithm\*\*: Applies the GrabCut algorithm which separates the foreground (main object) from the background using the rectangle.  
5. \*\*Create Final Mask\*\*: Pixels classified as background or probable background are set to 0 (black), others to 1 (white).  
6. \*\*Apply Mask\*\*: The original image is multiplied by the mask to extract only the foreground.  
7. \*\*Display and Save Output\*\*:  
 - Displays the result using `cv2.imshow`.  
 - Saves the processed image as "processed\_output.jpg".

**Output**

****