Name : Angga Agustian

Class : TI22T

SID : 20220040114

Add Score Project

Edge Detection Displaying the Contours of Objects in the Image Using an Edge Detection Algorithm.

1. Create a New Notebook in Google Colab
2. Install Required Libraries :
   !pip install opencv-python-headless numpy matplotlib
3. Upload image :
   from google.colab import files
   uploaded = files.upload()

4. Run Edge Detection Code :
   I. FULL CODE :

```
!pip install opencv-python-headless numpy matplotlib

from google.colab import files
uploaded = files.upload()

import cv2
import numpy as np
import matplotlib.pyplot as plt

def detect_edges(image_path):
    image = cv2.imread(image_path)
    if image is None:
        print("Error: Image not found.")
        return

    gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

    edges = cv2.Canny(gray_image, 50, 150)

    contours, _ = cv2.findContours(edges, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)

    result_image = image.copy()

    for contour in contours:
        cv2.drawContours(result_image, [contour], -1, (0, 255, 0), 2)

    print(f"Number of detected objects: {len(contours)}")

    plt.figure(figsize=(10, 5))
    plt.subplot(1, 2, 1)
    plt.title('Original Image')
    plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
    plt.axis('off')

    plt.subplot(1, 2, 2)
    plt.title('Detected Edges and Contours')
    plt.imshow(cv2.cvtColor(result_image, cv2.COLOR_BGR2RGB))
    plt.axis('off')

    plt.show()


image_path = list(uploaded.keys())[0]
detect_edges(image_path)
```

II. EXPLAINATION :

a. Imports :

import cv2
import numpy as np
import matplotlib.pyplot as plt

- **cv2** : For Computer Vision task (OpenCV)
- **numpy** : for numerical operations
- **matplotlib.pyplot** : for display images

b. Function Definition :

def detect_edges(image_path):

- **detect_edges** make the **image_path** as an argument

c. Image Reading :

```
image = cv2.imread(image_path)
if image is None:
    print("Error: Image not found.")
    return
```

- Reads the image from provided path
- If the image can't be found, it'll print an error message

d. Grayscale Conversion :

gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

- Converts the original image from BGR (default in OpenCV) to grayscale. This simplifies the edge detection process.

e. Edge Detection :

edges = cv2.Canny(gray_image, 50, 150)

- Applies the Canny edge detection algorithm. The parameters 50 and 150 set the lower and upper thresholds for edge detection.

f. Finding Contours :

contours, _ = cv2.findContours(edges, cv2.RETR_EXTERNAL,.CHAIN_APPROX_SIMPLE)

- Finds contours in the edge-detected image. The parameters specify that only external contours are retrieved, and the approximation method simplifies the contour representation.

g. Copying Original Image :

result_image = image.copy()

- Creates a copy of the original image where the detected contours will be drawn.

h. Drawing Countours :

```
for contour in contours:
    cv2.drawContours(result_image, [contour], -1, (0, 255, 0), 2)
```

- Loops through the detected contours and draws them on the **result_image** in green with a thickness of **2**.

i. Display Detected Objects :

print(f"Number of detected objects: {len(contours)}")

- Prints the number of objects (contours) detected in the image.

j. Image Display :

```
plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
plt.title('Original Image')
```

plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
plt.axis('off')

plt.subplot(1, 2, 2)
plt.title('Detected Edges and Contours')
plt.imshow(cv2.cvtColor(result_image, cv2.COLOR_BGR2RGB))
plt.axis('off')

plt.show()

- Creates a figure with two subplots: one showing the original image and the other showing the image with detected edges and contours. The images are converted from BGR to RGB for proper color representation in matplotlib.

k. Example Usage :
image_path = list(uploaded.keys())[0]
detect_edges(image_path)

- This part retrieves the first uploaded image's name from a dictionary (assumed to be available) and invokes the **detect_edges** function with that image.

5. Output :