OpenCV Image Processing

M.Junaid 109(4B)

Table of Contents

Introduction

This project demonstrates various image processing techniques using OpenCV, a powerful computer vision library. The implementation covers basic operations like image reading and display, as well as more advanced techniques including edge detection, morphological operations, and face detection.

Key Functionalities Implemented

1. Basic Image Operations

- **Image Loading and Display**: The project begins by loading an image ('pic.jpg') and displaying it using both OpenCV's native functions and Matplotlib.
- **Color Conversion**: Conversion from BGR to grayscale is demonstrated, which is fundamental for many computer vision tasks.
- **Image Resizing**: The image is resized to 200×200 pixels, showing how to adjust image dimensions.

2. Drawing and Annotation

- **Shape Drawing**: The code demonstrates drawing rectangles, circles, and lines on an image copy.
- **Text Annotation**: Text ("OpenCV") is added to the image using different font settings.

3. Image Processing Techniques

- **Thresholding**: Binary thresholding is applied to create a black-and-white version of the grayscale image.
- **Edge Detection**: The Canny edge detector is used to highlight edges in the image.
- **Blurring**: Gaussian blur is applied to smooth the image.
- **Morphological Operations**: Dilation, erosion, opening, and closing operations are demonstrated using a 5×5 kernel.
- **Gradient Calculation**: Sobel operators are used to compute image gradients in x and y directions.

4. Advanced Features

- **Contour Detection**: Contours are found in the thresholded image and drawn on the original image.
- **Face Detection**: Using Haar cascades, faces are detected in the image and marked with rectangles.

- **Webcam Feed**: A real-time webcam feed is displayed, which can be exited by pressing 'q'.
- **Bitwise Operations**: A bitwise AND operation is performed using the thresholded image as a mask.
- **Histogram Equalization**: The grayscale image's contrast is enhanced using histogram equalization.

Implementation Details

Dependencies

- OpenCV (cv2)
- NumPy
- Matplotlib

Key Code Snippets

1. **Image Loading and Display**:

   python

   Copy

   Download

   ```
   img = cv2.imread('pic.jpg')
   cv2.imshow('Image', img)
   cv2.waitKey(0)
   cv2.destroyAllWindows()
   ```

2. **Face Detection**:
   python

   Copy

   Download

```
face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haa
rcascade_frontalface_default.xml')
faces = face_cascade.detectMultiScale(gray, 1.1, 4)
```

3. **Webcam Feed**:

python

Copy

Download

```
cap = cv2.VideoCapture(0)
while True:
    ret, frame = cap.read()
    cv2.imshow('Webcam', frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
```

Results and Outputs

Grayscale Image

Resized Image

Shapes

Binary Threshold

Canny Edges

Dilation

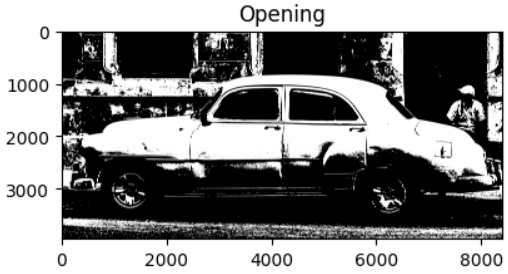
Sobel Gradient

Contours

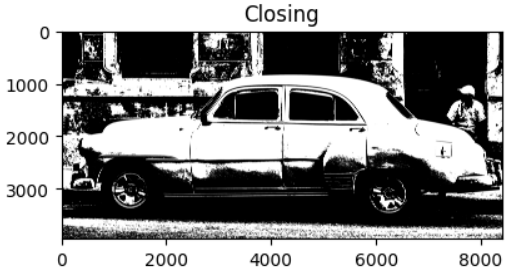
Bitwise AND

Erosion

Opening



Closing



Equalized Histogram