Day 2 Assignment: Circle Detector

Hough Transform Implementation
Computer Vision Bootcamp

made with ♡, by Aryan

Due: Next Session

1 Objective

Build a robust circle detection program using the Hough Circle Transform that can identify, analyze, and visualize circular objects in images.

2 Background

The Hough Circle Transform is a feature extraction technique used to detect circles in images. It's widely used in:

- Quality control (detecting circular parts)
- Medical imaging (identifying cells, tumors)
- Traffic sign detection
- Coin counting and classification
- Industrial inspection

2.1 How It Works

The Hough Circle Transform uses a voting procedure in a 3D parameter space (x, y, r) where:

- (x, y) is the circle center
- r is the radius

For each edge point, the algorithm considers all possible circles that could pass through it.

3 Algorithm Steps

3.1 Preprocessing

1. Convert image to grayscale
2. Apply Gaussian blur to reduce noise
3. (Optional) Enhance contrast with histogram equalization

3.2 Detection

1. Apply Hough Circle Transform with parameters:
    - `dp`: Inverse ratio of accumulator resolution
    - `minDist`: Minimum distance between circle centers
    - `param1`: Upper threshold for Canny edge detector
    - `param2`: Accumulator threshold (lower = more circles)
    - `minRadius`, `maxRadius`: Size constraints

3.3 Post-processing

1. Filter circles by confidence
2. Remove duplicate detections
3. Calculate statistics

4 Requirements

4.1 Functional Requirements

Your program must:

1. **Load and preprocess image**
    - Accept image file path as input
    - Convert to grayscale
    - Apply appropriate blur
2. **Detect circles**
    - Use `cv2.HoughCircles`
    - Handle cases with no circles found
    - Support parameter tuning
3. **Visualize results**
    - Draw circle outlines (green)
    - Mark circle centers (red)
    - Label each circle with ID and radius
    - Display original and annotated side-by-side
4. **Report statistics**
    - Total number of circles detected
    - Minimum, maximum, and average radius
    - List all circles with coordinates and radii
5. **Save results**
    - Save annotated image
    - Export statistics to text file

4.2 Code Quality

- Use functions (minimum 3 functions)
- Include docstrings for all functions
- Add parameter validation
- Handle errors gracefully
- Follow PEP 8 naming conventions

4.3 Testing

Test with at least 3 images:

1. Coins or circular objects
2. Image with overlapping circles
3. Challenging image (noise, varying sizes)

5 Starter Code

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

def preprocess_image(image_path):
    """
    Load and preprocess image for circle detection.

    Args:
        image_path: Path to input image

    Returns:
        tuple: (original_color, preprocessed_gray) or (None, None)
    """
    # TODO: Implement preprocessing
    pass


def detect_circles(gray_image, dp=1, minDist=50, param1=50,
                   param2=30, minRadius=10, maxRadius=100):
    """
    Detect circles using Hough Circle Transform.

    Args:
        gray_image: Preprocessed grayscale image
        dp: Inverse accumulator resolution ratio
        minDist: Minimum distance between circle centers
        param1: Upper Canny threshold
        param2: Accumulator threshold
        minRadius: Minimum circle radius
        maxRadius: Maximum circle radius

    Returns:
        numpy array of circles (x, y, radius) or None
```

```
    """
    # TODO: Apply HoughCircles
    pass


def visualize_circles(image, circles, save_path=None):
    """
    Draw detected circles on image and display.

    Args:
        image: Original color image
        circles: Array of detected circles
        save_path: Optional path to save annotated image
    """
    # TODO: Draw circles and labels
    pass


def calculate_statistics(circles):
    """
    Calculate and display statistics about detected circles.

    Args:
        circles: Array of detected circles

    Returns:
        dict: Statistics dictionary
    """
    # TODO: Compute statistics
    pass


def main():
    """Main function."""
    # TODO: Implement main workflow
    pass


if __name__ == '__main__':
    main()
```

6 Grading Rubric

Criteria	Points
Preprocessing Implementation	15
Circle Detection (Hough Transform)	25
Visualization Quality	20
Statistics Calculation	15
Code Quality & Organization	15
Testing (3 images)	10
Total	**100**

7 Bonus Challenges

7.1 Bonus 1: Parameter Auto-Tuning (+15 points)

Implement automatic parameter selection that adapts to different images.

7.2 Bonus 2: Interactive GUI (+20 points)

Create a GUI with sliders to adjust Hough parameters in real-time.

7.3 Bonus 3: Video Processing (+15 points)

Extend to detect circles in video frames and track them over time.

7.4 Bonus 4: Size Classification (+10 points)

Classify detected circles into size categories (small/medium/large) and color-code them.

8 Tips

- Start with `param2=30`, adjust based on results
- Lower `param2` if missing circles
- Higher `param2` if detecting false circles
- Use `minDist` to prevent duplicate detections
- Blur is crucial - test kernel sizes 3, 5, 7, 9
- For overlapping circles, decrease `minDist`

9 Submission

Create `Day2_Assignment_YourName.zip` containing:

```
Day2_Assignment_YourName/
|-- circle_detector.py
|-- test_images/
|   |-- test1.jpg
|   |-- test2.jpg
|   +-- test3.jpg
|-- results/
|   |-- result1.jpg
|   |-- result2.jpg
|   |-- result3.jpg
|   +-- statistics.txt
+-- README.txt
```

Good luck with your circle detection!