Project Report: Image Enhancement for Selected Poor Quality Images

Introduction

This report details the enhancement of poor-quality images using classical computer vision techniques in Python with OpenCV. The project is divided into four tasks:

1. **Component Extraction**: Extract circles and COVID-19 charts from specified images.
2. **Enhancement of Blurry Images**: Enhance blurry images of buildings and a dog.
3. **Noise Removal from Noisy Images**: Reduce noise in images containing text, a rocket, and a wind chart.
4. **Visual Enhancement of Challenging Images**: Improve the visual quality of a newspaper and a name plate.

For each task, we discuss the techniques used, provide the Python code, and analyze the results, including pros, cons, quality achieved, failure reasons, and suggestions for future processing.

Task Assignments Based on Project Images

The project specifies the following images for each task:

- **Component Extraction**: "Circle" and "COVID-19 Chart"
- **Enhancement of Blurry Images**: "Buildings" and "Dog"
- **Noise Removal**: "Text", "Rocket", and "Wind Chart"
- **Visual Enhancement**: "Newspaper" and "Name Plate"

[1] Component Extraction

Images: "Circle" and "COVID-19 Chart"

Techniques Used and Rationale

- **Gaussian Blur**:
  - **Why**: Reduces noise and detail, making it easier to detect shapes.
- **Hough Circle Transform** (for circles):
  - **Why**: Effective for detecting circular shapes, robust to noise and partial occlusions.
- **Contour Detection with Thresholding** (for chart):
  - **Why**: Thresholding separates the chart from the background, and contour detection isolates it as a distinct component.

Python Code

```
import cv2
import numpy as np

img_circle = cv2.imread('media/7 circles.png', 0)
img_chart = cv2.imread('media/Multi_objects_separation.jpeg', 0)

blurred_circle = cv2.GaussianBlur(img_circle, (5, 5), 0)
circles = cv2.HoughCircles(blurred_circle, cv2.HOUGH_GRADIENT, dp=1, minDist=20,
                           param1=50, param2=30, minRadius=0, maxRadius=0)
if circles is not None:
    circles = np.round(circles[0, :]).astype("int")
    for (x, y, r) in circles:
        cv2.circle(img_circle, (x, y), r, (255), 4)

blurred_chart = cv2.GaussianBlur(img_chart, (5, 5), 0)
thresh = cv2.adaptiveThreshold(blurred_chart, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C, cv2.THRESH_BI
contours, _ = cv2.findContours(thresh, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
min_area = 100
for c in contours:
    if cv2.contourArea(c) > min_area:
        x, y, w, h = cv2.boundingRect(c)
        cv2.rectangle(img_chart, (x, y), (x + w, y + h), (255), 2)

cv2.imwrite('output_image_circle.jpg', img_circle)
cv2.imwrite('output_image_chart.jpg', img_chart)
```

Output

- **Circle Image**: Circles highlighted with outlines.
- **Chart Image**: Chart area enclosed in a bounding rectangle.

Discussion

- **Pros**: Accurate detection of circles and chart regions.
- **Cons**: Noise or low contrast may lead to missed detections.
- **Quality Achieved**: Good for prominent components, less effective for faint details.
- **Failure Reasons**: Insufficient contrast or overlapping objects.
- **Suggestions**: Pre-process with contrast enhancement (e.g., histogram equalization).

[2] Enhancement of Blurry Images

Images: "Buildings" and "Dog"

Techniques Used and Rationale

- **Gaussian Blur**:
  - **Why**: Reduces noise and detail, making it easier to detect shapes.
- **Bilateral Filter**:
  - **Why**: Preserves edges while smoothing the image, reducing blur.
- **Unsharp Masking**:
  - **Why**: Enhances edges and details, making the image appear sharper.
- **Adaptive Histogram Equalization**:
  - **Why**: Improves local contrast, enhancing visibility in low-contrast areas.
- **CLAHE (Contrast Limited Adaptive Histogram Equalization)**:
  - **Why**: Prevents over-amplification of noise while enhancing contrast.
- **Color Space Conversion**:
  - **Why**: Converts to HSV for better control over brightness and contrast.
- **Equalization**:
  - **Why**: Enhances the brightness channel to improve overall visibility.
- **Histogram Equalization**:
  - **Why**: Improves global contrast, enhancing overall visibility.
- **Sharpening**:
  - **Why**: Enhances edges to reduce blur.

Python Code

```python
import cv2
import numpy as np

img_building = cv2.imread('media/building.jpg')
img_dog = cv2.imread('media/dog.jpg')

def enhance_image(img):
    imhsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
    imhsv[:, :, 2] = cv2.equalizeHist(imhsv[:, :, 2])
    img_enhanced = cv2.cvtColor(imhsv, cv2.COLOR_HSV2BGR)
    kernel = np.array([[0, -1, 0], [-1, 5, -1], [0, -1, 0]])
    img_sharpened = cv2.filter2D(img_enhanced, -1, kernel)
    return img_sharpened

img_building_enhanced = enhance_image(img_building)
img_dog_enhanced = enhance_image(img_dog)

cv2.imwrite('output_image_building.jpg', img_building_enhanced)
cv2.imwrite('output_image_dog.jpg', img_dog_enhanced)
```

Output

- **Buildings Image**: Improved contrast and sharper edges.
- **Dog Image**: Enhanced features with reduced blur.

Discussion

- **Pros**: Noticeable improvement in clarity and detail.
- **Cons**: Possible over-sharpening artifacts.
- **Quality Achieved**: Visibly sharper images.
- **Failure Reasons**: Severe blur may not be fully corrected.
- **Suggestions**: Explore deblurring filters (e.g., Wiener).

[3] Noise Removal from Noisy Images

Images: "Text", "Rocket", and "Wind Chart"

Techniques Used and Rationale

- **Non-Local Means Denoising**:
  - **Why**: Reduces noise while preserving edges and details.

Python Code

```
import cv2
import numpy as np

img_text = cv2.imread('media/text.jpg', 0)
img_rocket = cv2.imread('media/rocket.jpg', 0)
img_wind_chart = cv2.imread('media/wind_chart.jpg', 0)

def denoise_image(img):
    img_bgr = cv2.cvtColor(img, cv2.COLOR_GRAY2BGR)
    img_denoised = cv2.fastNlMeansDenoisingColored(img_bgr, None, 10, 10, 7, 21)
    return cv2.cvtColor(img_denoised, cv2.COLOR_BGR2GRAY)

img_text_denoised = denoise_image(img_text)
img_rocket_denoised = denoise_image(img_rocket)
img_wind_chart_denoised = denoise_image(img_wind_chart)

cv2.imwrite('output_image_text.jpg', img_text_denoised)
cv2.imwrite('output_image_rocket.jpg', img_rocket_denoised)
cv2.imwrite('output_image_wind_chart.jpg', img_wind_chart_denoised)
```

Output

- **Text Image**: Clearer text with reduced noise.
- **Rocket Image**: Cleaner image with better contrast.
- **Wind Chart Image**: Improved readability.

Discussion

- **Pros**: Effective noise reduction with detail preservation.
- **Cons**: Slight blurring of fine details possible.

- **Quality Achieved**: Readable and cleaner images.
- **Failure Reasons**: High noise levels may persist.
- **Suggestions**: Combine with median filtering for specific noise types.

[4] Visual Enhancement of Challenging Images

Images: "Newspaper" and "Name Plate"

Techniques Used and Rationale

- **Histogram Equalization**:
  - **Why**: Enhances contrast in low-visibility areas.
- **Sharpening**:
  - **Why**: Improves text sharpness and readability.

Python Code

```
import cv2
import numpy as np

img_newspaper = cv2.imread('media/Newspaper.jpg', 0)
img_nameplate = cv2.imread('media/Name plate.jpg', 0)

img_newspaper = cv2.equalizeHist(img_newspaper)
img_nameplate = cv2.equalizeHist(img_nameplate)

kernel = np.array([[0, -1, 0], [-1, 5, -1], [0, -1, 0]])
img_newspaper_enhanced = cv2.filter2D(img_newspaper, -1, kernel)
img_nameplate_enhanced = cv2.filter2D(img_nameplate, -1, kernel)

cv2.imwrite('output_image_newspaper.jpg', img_newspaper_enhanced)
cv2.imwrite('output_image_nameplate.jpg', img_nameplate_enhanced)
```

Output

- **Newspaper Image**: Improved text readability.
- **Name Plate Image**: Clearer text and details.

Discussion

- **Pros**: Significant enhancement of text visibility.
- **Cons**: Risk of noise amplification.
- **Quality Achieved**: Readable text and improved quality.
- **Failure Reasons**: Severe degradation may limit results.
- **Suggestions**: Apply noise reduction pre-processing.

Good Luck!