Lesson 1: Image Filters and Blurring

Learning Objective:

Understand how image filtering works using convolution and built-in OpenCV blurring methods.

Topics:

- What is an image filter?
- Gaussian Blur
- Median Blur
- Bilateral Filter

Example Code:

```
import cv2

img = cv2.imread('sample.jpg')

# Apply Gaussian Blur
gaussian = cv2.GaussianBlur(img, (9, 9), 0)

# Apply Median Blur
median = cv2.medianBlur(img, 5)

# Apply Bilateral Filter
bilateral = cv2.bilateralFilter(img, 9, 75, 75)

cv2.imshow('Gaussian', gaussian)
cv2.imshow('Median', median)
cv2.imshow('Bilateral', bilateral)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Task 1: Apply and Compare Filters

Instructions for the Intern:

- Load `sample.jpg`.

- Apply all three filters (Gaussian, Median, Bilateral).
- Save each result with filenames: `gaussian.jpg`, `median.jpg`, `bilateral.jpg`.
- Write a short note comparing the visual effects of each.

Lesson 2: Custom Image Filters with Kernels

Learning Objective:

Learn to apply custom filters like sharpening and edge detection using convolution kernels.

Topics:

- Convolution basics
- cv2.filter2D
- Sharpening kernel
- Edge detection kernel

Example Code:

```
import cv2
import numpy as np

img = cv2.imread('sample.jpg')

# Sharpening kernel
kernel_sharpen = np.array([[0, -1, 0],
                    [-1, 5,-1],
                    [0, -1, 0]])

# Edge detection kernel
kernel_edge = np.array([[-1, -1, -1],
                  [-1, 8, -1],
                  [-1, -1, -1]])

sharpened = cv2.filter2D(img, -1, kernel_sharpen)
edges = cv2.filter2D(img, -1, kernel_edge)

cv2.imshow('Sharpened', sharpened)
cv2.imshow('Edges', edges)
```

```
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Task 2: Create Your Own Filters

Instructions for the Intern:

- Load an image and apply both the sharpening and edge detection kernels.
- Try changing values in the kernel to create a custom effect.
- Save the outputs: `sharpened.jpg`, `edges.jpg`, `custom_filter.jpg`.
- Write 2-3 lines describing the effect of the custom filter.

Lesson 3: Image Arithmetic & Bitwise Operations

Learning Objective:

Understand how to blend images and use bitwise operations for masking.

Topics:

- Image addition and blending
- Bitwise AND, OR, NOT operations
- Creating masks

Example Code (Blending):

```
import cv2

img1 = cv2.imread('sample.jpg')
img2 = cv2.imread('logo.jpg')  # smaller image

# Resize img2 to match img1
img2 = cv2.resize(img2, (img1.shape[1], img1.shape[0]))

# Blend the two
blended = cv2.addWeighted(img1, 0.7, img2, 0.3, 0)

cv2.imshow('Blended', blended)
```

```
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Example Code (Bitwise):

```
import cv2
import numpy as np

img = cv2.imread('sample.jpg')
mask = np.zeros(img.shape[:2], dtype="uint8")
cv2.rectangle(mask, (50, 50), (200, 200), 255, -1)

masked = cv2.bitwise_and(img, img, mask=mask)

cv2.imshow("Masked Region", masked)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Task 3: Blend and Mask

Instructions for the Intern:

- Blend two images together using `cv2.addWeighted`.
- Create a mask with a rectangle or circle.
- Apply a bitwise AND operation to show only the masked region.
- Save the output as `blended.jpg` and `masked.jpg`.

Great! Let’s continue building **Module 2: Intermediate Computer Vision with OpenCV** with the next set of lessons.

Lesson 4: Perspective and Affine Transformations

Learning Objective:

Understand how to manipulate the orientation and shape of images using geometric transforms

Topics:

- Difference between Affine and Perspective transforms
- `cv2.getAffineTransform()` and `cv2.warpAffine()`
- `cv2.getPerspectiveTransform()` and `cv2.warpPerspective()`

Example Code: Affine Transform

```
import cv2
import numpy as np

img = cv2.imread('sample.jpg')

# Define source and destination points
pts1 = np.float32([[50, 50], [200, 50], [50, 200]])
pts2 = np.float32([[10, 100], [200, 50], [100, 250]])

matrix = cv2.getAffineTransform(pts1, pts2)
result = cv2.warpAffine(img, matrix, (img.shape[1], img.shape[0]))

cv2.imshow("Affine Transform", result)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Example Code: Perspective Transform

```
import cv2
import numpy as np

img = cv2.imread('document.jpg')

# Define 4 corner points of the document in the image
pts1 = np.float32([[100, 100], [500, 100], [100, 400], [500, 400]])
pts2 = np.float32([[0, 0], [300, 0], [0, 400], [300, 400]])

matrix = cv2.getPerspectiveTransform(pts1, pts2)
result = cv2.warpPerspective(img, matrix, (300, 400))

cv2.imshow("Warped Image", result)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Task 4: Apply Geometric Transforms

Instructions for the Intern:

- Take any sample image.
- Apply an affine transformation by shifting and rotating a triangle region.
- Apply a perspective transformation to simulate a "document scan".
- Save results as `affine_result.jpg` and `perspective_result.jpg`.

Lesson 5: Image Pyramids & Zooming

Learning Objective:

Use pyramids to create scaled versions of an image for zoom effects and multiscale analysis.

Topics:

- Gaussian pyramid (downsampling)
- Laplacian pyramid (edge emphasis)
- `cv2.pyrDown()` and `cv2.pyrUp()`

Example Code:

```
import cv2

img = cv2.imread('sample.jpg')

lower_reso = cv2.pyrDown(img)
higher_reso = cv2.pyrUp(lower_reso)

cv2.imshow("Original", img)
cv2.imshow("Downsampled", lower_reso)
cv2.imshow("Upsampled", higher_reso)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Task 5: Build and Compare Pyramids

Instructions for the Intern:

- Load `sample.jpg`.
- Apply `cv2.pyrDown()` twice and then `cv2.pyrUp()` on the result.
- Save the images: `down1.jpg`, `down2.jpg`, `up_from_down2.jpg`.
- Compare and describe any quality loss during this process.

Lesson 6: Trackbars for Real-Time Parameters

Learning Objective:

Use OpenCV GUI elements to adjust image processing parameters in real-time.

Topics:

- `cv2.createTrackbar()`
- Real-time Canny thresholding
- Custom filters with trackbars

Example Code:

```
import cv2

def nothing(x):
    pass

img = cv2.imread('sample.jpg', 0)
cv2.namedWindow('Canny')

# Create trackbars for thresholds
cv2.createTrackbar('Min', 'Canny', 0, 255, nothing)
cv2.createTrackbar('Max', 'Canny', 0, 255, nothing)

while True:
    min_val = cv2.getTrackbarPos('Min', 'Canny')
    max_val = cv2.getTrackbarPos('Max', 'Canny')

    edges = cv2.Canny(img, min_val, max_val)
    cv2.imshow('Canny', edges)
    if cv2.waitKey(1) & 0xFF == 27:
        break

cv2.destroyAllWindows()
```

Task 6: Interactive Edge Detector

Instructions for the Intern:

- Use trackbars to control the min and max thresholds of Canny edge detection.
- Save a screenshot of your favorite output as `tracked_canny.jpg`.
- Briefly explain how threshold values affect the edge output.

Perfect! Let’s now continue with the final lessons of **Module 2: Intermediate Computer Vision with OpenCV (Part 3)**.

Lesson 7: Image Histograms & Contrast Enhancement

Learning Objective:

Understand image histograms and apply techniques like histogram equalization to improve contrast.

Topics:

- Histogram plotting with `cv2.calcHist()`
- Histogram equalization with `cv2.equalizeHist()`
- CLAHE (Contrast Limited Adaptive Histogram Equalization)

Example Code:

```
import cv2

import matplotlib.pyplot as plt



img = cv2.imread('sample.jpg', 0)
```

```
# Plot histogram
hist = cv2.calcHist([img], [0], None, [256], [0, 256])
plt.plot(hist)
plt.show()

# Equalize
equalized = cv2.equalizeHist(img)

cv2.imshow('Original', img)
cv2.imshow('Equalized', equalized)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Task 7: Histogram Analysis & Enhancement

Instructions for the Intern:

- Load a grayscale image.
- Plot and save its histogram using Matplotlib (`histogram.png`).
- Apply `cv2.equalizeHist()` and CLAHE (adaptive version).
- Save enhanced images: `equalized.jpg`, `clahe.jpg`.

Lesson 8: Morphological Transformations

Learning Objective:

Learn to apply operations like erosion, dilation, opening, and closing to clean up binary images.

Topics:

- Erosion and Dilation
- Opening and Closing
- `cv2.getStructuringElement()`

Example Code:

```
import cv2
import numpy as np

img = cv2.imread('binary.png', 0)
kernel = np.ones((5, 5), np.uint8)

erosion = cv2.erode(img, kernel, iterations=1)
dilation = cv2.dilate(img, kernel, iterations=1)
opening = cv2.morphologyEx(img, cv2.MORPH_OPEN, kernel)
closing = cv2.morphologyEx(img, cv2.MORPH_CLOSE, kernel)

cv2.imshow("Erosion", erosion)
cv2.imshow("Dilation", dilation)
cv2.imshow("Opening", opening)
cv2.imshow("Closing", closing)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Task 8: Morphological Processing

Instructions for the Intern:

- Use a binary image with noise.
- Apply erosion, dilation, opening, and closing.
- Save results: `erosion.jpg`, `dilation.jpg`, `opening.jpg`, `closing.jpg`.
- Briefly explain the differences.

Lesson 9: Template Matching

Learning Objective:

Detect smaller objects or logos inside an image using template matching.

Topics:

- `cv2.matchTemplate()`
- `cv2.minMaxLoc()`

Example Code:

```
import cv2


img = cv2.imread('scene.jpg')
template = cv2.imread('object.jpg')


res = cv2.matchTemplate(img, template, cv2.TM_CCOEFF_NORMED)
min_val, max_val, min_loc, max_loc = cv2.minMaxLoc(res)


(h, w) = template.shape[:2]
cv2.rectangle(img, max_loc, (max_loc[0] + w, max_loc[1] + h), (0, 255, 0), 2)
```

```
cv2.imshow("Detected", img)

cv2.waitKey(0)

cv2.destroyAllWindows()
```

Task 9: Object Locator

Instructions for the Intern:

- Take a template image of a logo or small object.
- Use `cv2.matchTemplate()` to find it in a larger scene.
- Draw a rectangle around the best match.
- Save the result as `template_match.jpg`.

Mini Project: Shape Counter

Project Objective:

Count the number of geometric shapes (circles, squares, triangles) in an image using contours and classification logic.

Hints:

- Use `cv2.findContours()`
- Use `cv2.approxPolyDP()` to classify based on number of sides
- Use `cv2.putText()` to label each shape

Final Task: Shape Counter App

Instructions for the Intern:

- Create `shape_counter.py`
- Load a shape-rich image (circles, rectangles, etc.)
- Detect and label each shape (Triangle, Square, Circle, etc.)

- Print the count of each shape.
- Save the final annotated image as `shapes_detected.jpg`