Exercise 1 - Image Processing Techniques

September 16, 2024

1 Apply image processing techniques (Scaling, Rotation, Blurring, Edge Detection) using OpenCV

1.1 Step 1: Install OpenCV

[1]:
```
!pip install opencv-python-headless
```

```
Requirement already satisfied: opencv-python-headless in
/usr/local/lib/python3.10/dist-packages (4.10.0.84)
Requirement already satisfied: numpy>=1.21.2 in /usr/local/lib/python3.10/dist-
packages (from opencv-python-headless) (1.26.4)
```

OpenCV is a powerful library for image processing. The opencv-python-headless package is installed using the pip command in Google Colab. This version of OpenCV is designed for use in environments where GUI operations are not necessary, like in Colab.

1.2 Step 2: Import Necessary Libraries

[2]:
```python
import cv2
import numpy as np
import matplotlib.pyplot as plt

# Function to display an image using matplotlib
def display_image(img, title="Image"):
    plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
    plt.title(title)
    plt.axis('off')
    plt.show()

# Function to display two images side by side
def display_images(img1, img2, title1="Image 1", title2="Image 2"):
    plt.subplot(1, 2, 1)
    plt.imshow(cv2.cvtColor(img1, cv2.COLOR_BGR2RGB))
    plt.title(title1)
    plt.axis('off')

    plt.subplot(1, 2, 2)
    plt.imshow(cv2.cvtColor(img2, cv2.COLOR_BGR2RGB))
    plt.title(title2)
    plt.axis('off')

    plt.show()
```

- **cv2:** This imports OpenCV, which provides functions for image processing.
- **numpy (np):** This library is used for handling arrays and matrices, which images are represented as.
- **matplotlib.pyplot (plt):** This is used to display images in a Jupyter notebook or Google Colab environment.

1.3 Step 3: Load an Image

[3]:
```
from google.colab import files
from io import BytesIO
from PIL import Image

# Upload an image
uploaded = files.upload()

# Convert to OpenCV format
image_path = next(iter(uploaded))  # Get the image file name
image = Image.open(BytesIO(uploaded[image_path]))
image = cv2.cvtColor(np.array(image), cv2.COLOR_RGB2BGR)

display_image(image, "Original Image")
```

```
<IPython.core.display.HTML object>
```

```
Saving Orencia.png to Orencia.png
```

Original Image

- **display_image():** Converts the image from BGR (OpenCV's default color format) to RGB (the format expected by matplotlib) and displays it using imshow().
- **display_images():** This function allows two images to be displayed side by side for comparison. We use subplot to create a grid of plots (here, 1 row and 2 columns).

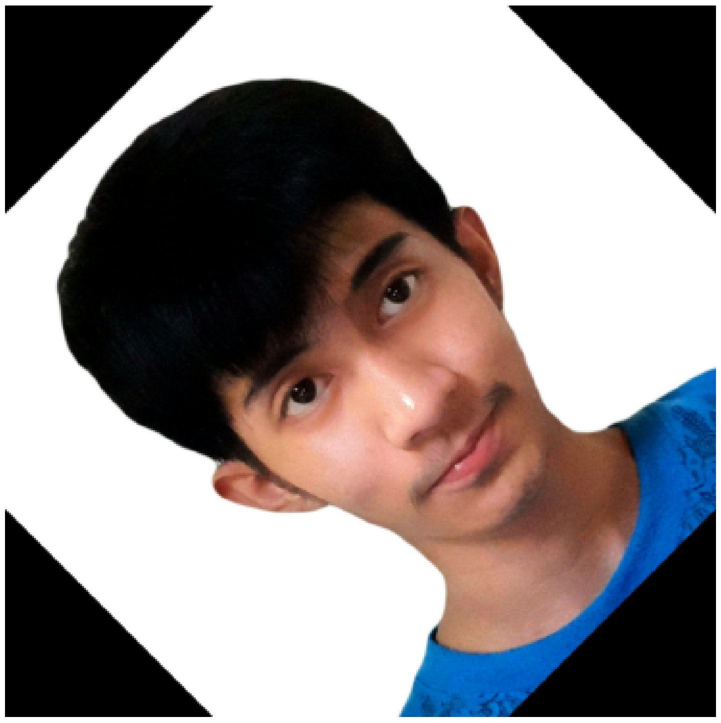
2 Exercise 1: Scaling and Rotation

[4]:
```
# Scaling
def scale_image(img, scale_factor):
    height, width = img.shape[:2]
    scaled_img = cv2.resize(img,
(int(width * scale_factor), int(height * scale_factor)), interpolation=cv2.
 ↪INTER_LINEAR)
    return scaled_img

"""
scale_image(): This function scales the image by a given factor.
The cv2.resize() function takes the original dimensions of the image,
multiplies them by the scale_factor, and resizes the image accordingly.
INTER_LINEAR is a common interpolation method for resizing.
"""
```

```
# Rotate
def rotate_image(img, angle):
    height, width = img.shape[:2]
    center = (width // 2, height // 2)
    matrix = cv2.getRotationMatrix2D(center, angle, 1.0)
    rotated_img = cv2.warpAffine(img, matrix, (width, height))
    return rotated_img

"""
rotate_image(): Rotates the image around its center. cv2.getRotationMatrix2D()␣
 ↪creates
a transformation matrix for rotation, and cv2.warpAffine() applies this␣
 ↪transformation.
The angle parameter controls the degree of rotation.
"""

# Scale image by 0.5
scaled_image = scale_image(image, 0.5)
display_image(scaled_image, "Scaled Image (50%)")

# Rotate image by 45 degrees
rotated_image = rotate_image(image, 45)
display_image(rotated_image, "Rotated Image (45°)")

"""
These lines apply the scaling and rotation functions to the uploaded image and␣
 ↪display the results.
"""
```

Scaled Image (50%)

Rotated Image (45°)



```
[4]: '\nThese lines apply the scaling and rotation functions to the uploaded image
     and display the results.\n'
```

3 Exercise 2: Blurring Techniques

```
[5]: # Gaussian Blur
     gaussian_blur = cv2.GaussianBlur(image, (5, 5), 0)
     display_image(gaussian_blur, "Gaussian Blur (5x5)")

     """
     cv2.GaussianBlur(): Applies a Gaussian blur to the image, which smooths it by␣
      ↪averaging
     the pixel values in a 5x5 kernel (a small matrix). This is useful for reducing␣
      ↪noise in an image.
     """

     # Median Blur
     median_blur = cv2.medianBlur(image, 5)
     display_image(median_blur, "Median Blur (5x5)")

     """
     cv2.medianBlur(): Applies a median blur, which replaces each pixel's value with␣
      ↪the
     median value of its neighbors in a 5x5 kernel. This method is particularly␣
      ↪effective in
     removing salt-and-pepper noise.
     """
```

Gaussian Blur (5x5)

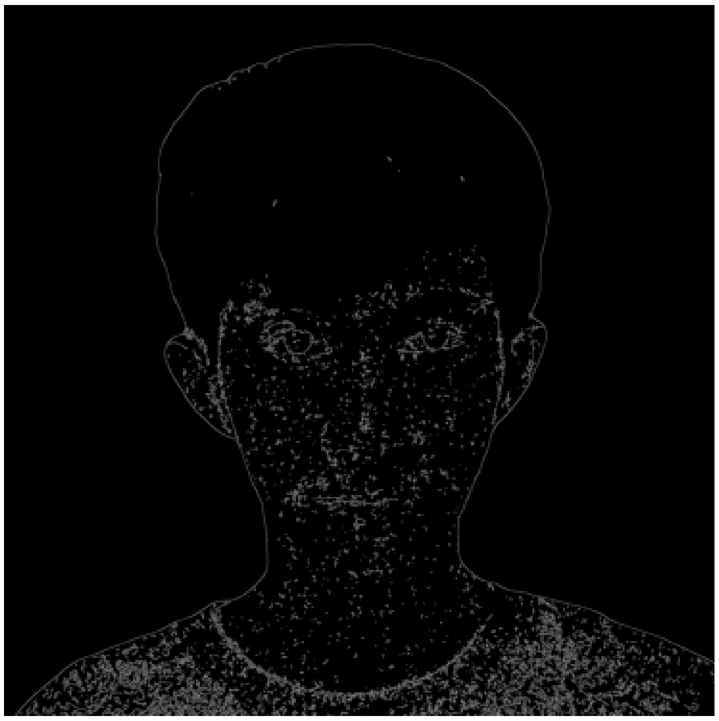
Median Blur (5x5)

```
[5]: "\ncv2.medianBlur(): Applies a median blur, which replaces each pixel's value
     with the\nmedian value of its neighbors in a 5x5 kernel. This method is
     particularly effective in\nremoving salt-and-pepper noise.\n"
```

4 Exercise 3: Edge Detection using Canny

```
[6]: # Canny Edge Detection
     edges = cv2.Canny(image, 100, 200)
     display_image(edges, "Canny Edge Detection (100, 200)")

     """
     cv2.Canny(): Detects edges in the image by calculating the gradient (rate of␣
      ↪intensity change)
     between pixels. The two threshold values (100 and 200) define the edges'
     sensitivity. Lower thresholds detect more edges, while higher thresholds detect␣
      ↪only the
     most prominent edges.
     """
```

Canny Edge Detection (100, 200)

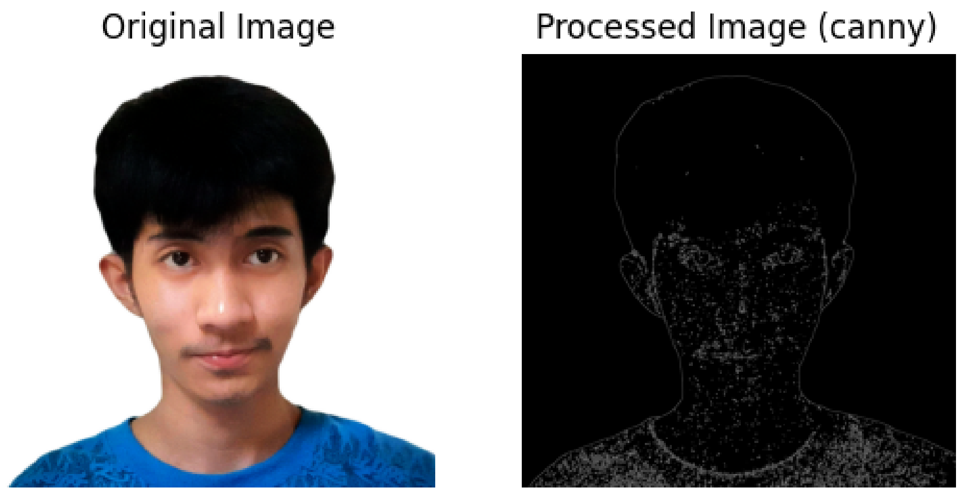


```
[6]: "\ncv2.Canny(): Detects edges in the image by calculating the gradient (rate of
     intensity change)\nbetween pixels. The two threshold values (100 and 200) define
     the edges'\nsensitivity. Lower thresholds detect more edges, while higher
     thresholds detect only the\nmost prominent edges.\n"
```

5 Exercise 4: Basic Image Processor (Interactive)

```
[7]: def process_image(img, action):
         if action == 'scale':
             return scale_image(img, 0.5)
         elif action == 'rotate':
             return rotate_image(img, 45)
         elif action == 'gaussian_blur':
             return cv2.GaussianBlur(img, (5, 5), 0)
         elif action == 'median_blur':
             return cv2.medianBlur(img, 5)
         elif action == 'canny':
             return cv2.Canny(img, 100, 200)
         else:
             return img

     """
     process_image(): This function allows users to specify an image transformation␣
      ↪(scaling,
     rotation, blurring, or edge detection). Depending on the action passed, it will␣
      ↪apply the
     corresponding image processing technique and return the processed image.
     """

     action = input("Enter action (scale, rotate, gaussian_blur, median_blur, canny):
      ↪ ")
     processed_image = process_image(image, action)
     display_images(image, processed_image, "Original Image", f"Processed Image␣
      ↪({action})")

     """
     This allows users to enter their desired transformation interactively (via the
     input() function). It processes the image and displays both the original and␣
      ↪transformed
     versions side by side.
     """
```

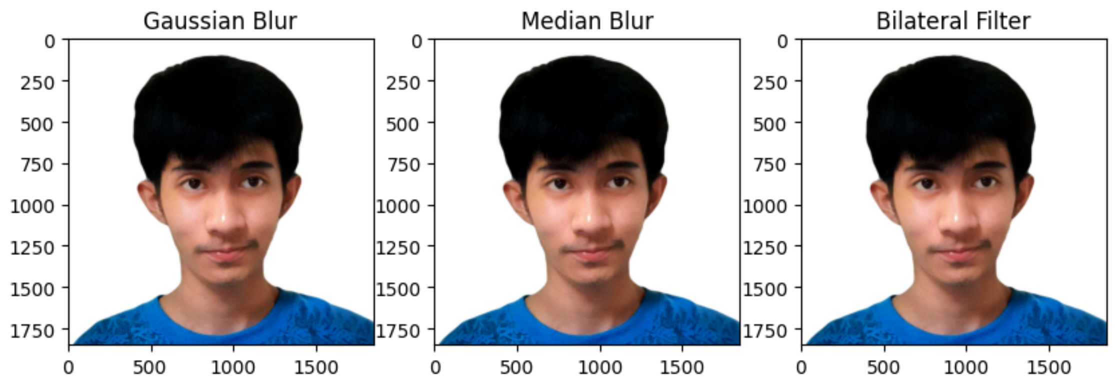
```
Enter action (scale, rotate, gaussian_blur, median_blur, canny): canny
```



[7]:
```
'\nThis allows users to enter their desired transformation interactively (via
the\ninput() function). It processes the image and displays both the original
and transformed\nversions side by side.\n'
```

6 Exercise 5: Comparison of Filtering Techniques

[8]:
```python
# Applying Gaussian, Median, and Bilateral filters
gaussian_blur = cv2.GaussianBlur(image, (5, 5), 0)
median_blur = cv2.medianBlur(image, 5)
bilateral_filter = cv2.bilateralFilter(image, 9, 75, 75)

"""
cv2.bilateralFilter(): This filter smooths the image while keeping edges sharp,␣
 ↪unlike
Gaussian or median filters. It's useful for reducing noise while preserving␣
 ↪details.
"""

# Display the results for comparison
plt.figure(figsize=(10, 5))
plt.subplot(1, 3, 1)
plt.imshow(cv2.cvtColor(gaussian_blur, cv2.COLOR_BGR2RGB))
plt.title("Gaussian Blur")

plt.subplot(1, 3, 2)
plt.imshow(cv2.cvtColor(median_blur, cv2.COLOR_BGR2RGB))
plt.title("Median Blur")

plt.subplot(1, 3, 3)
plt.imshow(cv2.cvtColor(bilateral_filter, cv2.COLOR_BGR2RGB))
plt.title("Bilateral Filter")

plt.show()

"""
Explanation: This displays the images processed by different filtering␣
  ↪techniques (Gaussian,
Median, and Bilateral) side by side for comparison.
"""
```

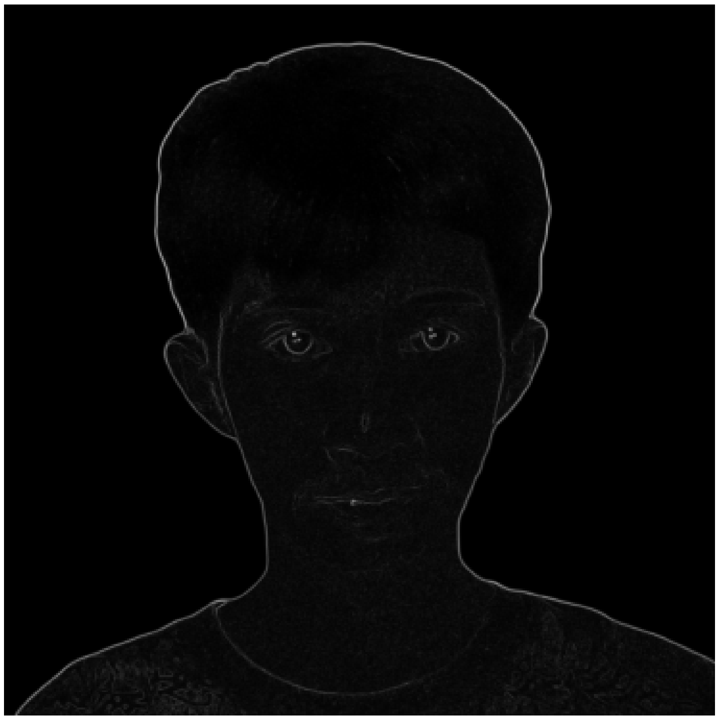


[8]:
```
'\nExplanation: This displays the images processed by different filtering
techniques (Gaussian,\nMedian, and Bilateral) side by side for comparison.\n'
```

7 Exercise 6: Sobel Edge Detection

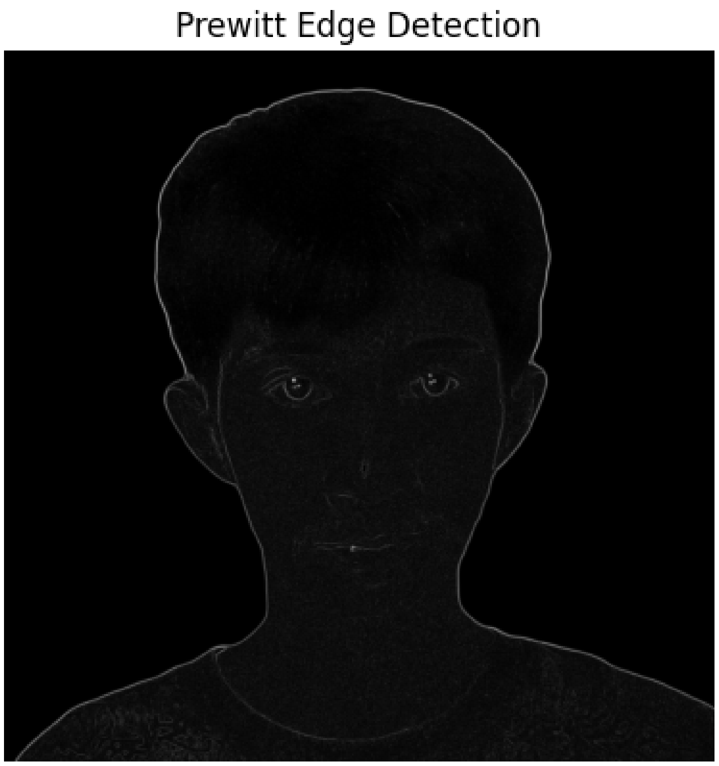
[9]:
```
def sobel_edge_detection(img):
  # Convert to grayscale
  gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
  # Sobel edge detection in the x direction
  sobelx = cv2.Sobel(gray, cv2.CV_64F, 1, 0, ksize=5)
  # Sobel edge detection in the y direction
  sobely = cv2.Sobel(gray, cv2.CV_64F, 0, 1, ksize=5)
  # Combine the two gradients
  sobel_combined = cv2.magnitude(sobelx, sobely)
  return sobel_combined

# Apply sobel edge detection to the uploaded image
sobel_edges = sobel_edge_detection(image)
plt.imshow(sobel_edges, cmap='gray')
plt.title("Sobel Edge Detection")
plt.axis('off')
plt.show()
```

Sobel Edge Detection



8 Exercise 7: Prewitt Edge Detection

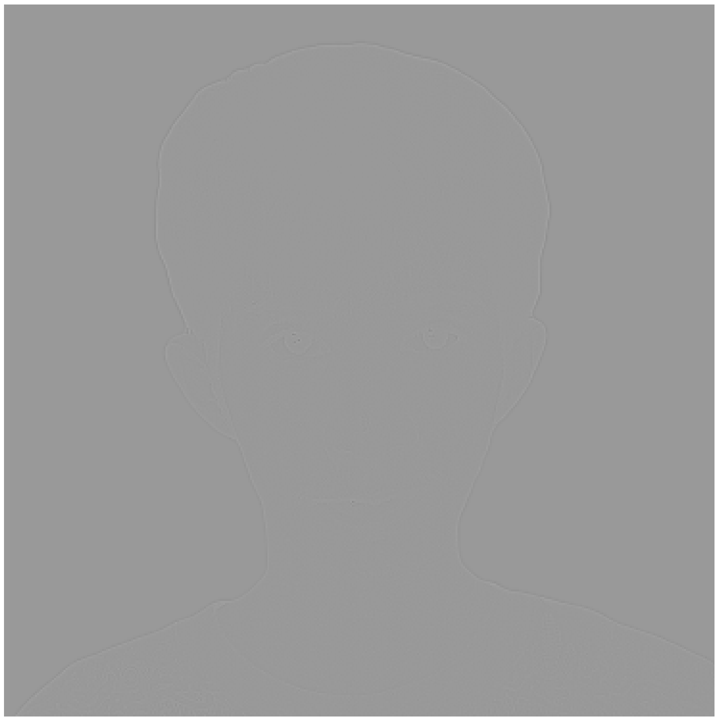
[10]:
```
def prewitt_edge_detection(img):
    # Convert to grayscale
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    # Prewitt operator kernels for x and y directions
    kernelx = np.array([[1, 0, -1], [1, 0, -1], [1, 0, -1]], dtype=int)
    kernely = np.array([[1, 1, 1], [0, 0, 0], [-1, -1, -1]], dtype=int)
    # Applying the Prewitt operator
    prewittx = cv2.filter2D(gray, cv2.CV_64F, kernelx)
    prewitty = cv2.filter2D(gray, cv2.CV_64F, kernely)
    # Combine the x and y gradients by converting to floating point
    prewitt_combined = cv2.magnitude(prewittx, prewitty)
    return prewitt_combined

# Apply Prewitt edge detection to the uploaded image
prewitt_edges = prewitt_edge_detection(image)
plt.imshow(prewitt_edges, cmap='gray')
plt.title("Prewitt Edge Detection")
plt.axis('off')
plt.show()
```



9 Exercise 8: Laplacian Edge Detection

[11]:
```
def laplacian_edge_detection(img):
  # Convert to grayscale
  gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
  # Apply Laplacian operator
  laplacian = cv2.Laplacian(gray, cv2.CV_64F)
  return laplacian

# Apply Laplacian edge detection to the uploaded image
laplacian_edges = laplacian_edge_detection(image)
plt.imshow(laplacian_edges, cmap='gray')
plt.title("Laplacian Edge Detection")
plt.axis('off')
plt.show()
```

Laplacian Edge Detection



10 Exercise 9: Bilateral Filter

[12]:
```
def bilateral_blur(img):
  bilateral = cv2.bilateralFilter(img, 9, 75, 75)
  return bilateral

# Apply Bilateral filter to the uploaded image
bilateral_blurred = bilateral_blur(image)
plt.imshow(cv2.cvtColor(bilateral_blurred, cv2.COLOR_BGR2RGB))
plt.title("Bilateral Filter")
plt.axis('off')
plt.show()
```

Bilateral Filter

11 Exercise 10: Box Filter

[13]:
```python
# Box Filter
def box_blur(img):
  box = cv2.boxFilter(img, -1, (5, 5))
  return box

# Apply Box filter to the uploaded image
box_blurred = box_blur(image)
plt.imshow(cv2.cvtColor(box_blurred, cv2.COLOR_BGR2RGB))
plt.title("Box Filter")
plt.axis('off')
plt.show()
```

Box Filter

12 Exercise 11: Motion Blur

[14]:
```python
def motion_blur(img):
    # Create motion blur kernel (size 15x15)
    kernel_size = 15
    kernel = np.zeros((kernel_size, kernel_size))
    kernel[int((kernel_size - 1) / 2), :] = np.ones(kernel_size)
    kernel = kernel / kernel_size

    # Apply motion blur
    motion_blurred = cv2.filter2D(img, -1, kernel)
    return motion_blurred

# Apply Motion blur to the uploaded image
motion_blurred = motion_blur(image)
plt.imshow(cv2.cvtColor(motion_blurred, cv2.COLOR_BGR2RGB))
plt.title("Motion Blur")
plt.axis('off')
plt.show()
```

Motion Blur

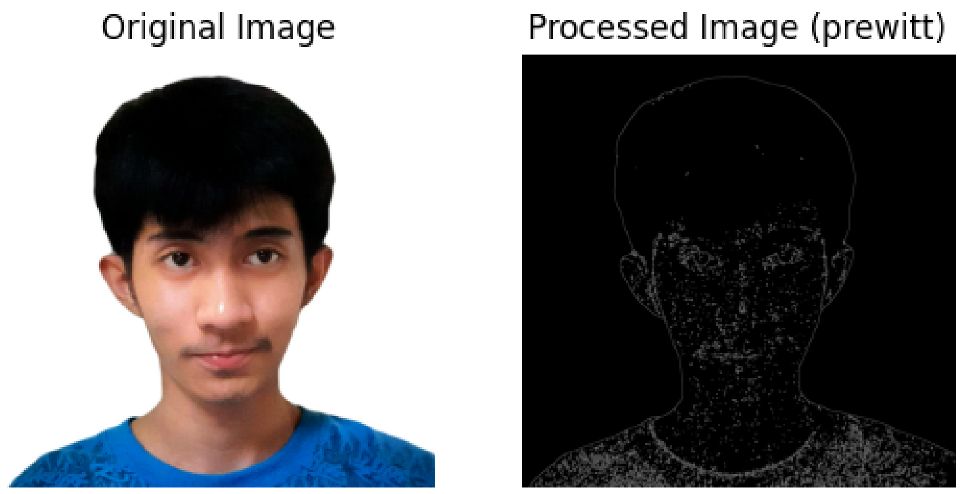
13 Exercise 12: Unsharp Masking (Sharpening)

```
[15]: def unsharp_mask(img):
        # Create a Gaussian blur version of the image
        blurred = cv2.GaussianBlur(img, (9, 9), 10.0)
        # Sharpen by adding the difference between the original and the blurred image
        sharpened = cv2.addWeighted(img, 1.5, blurred, -0.5, 0)
        return sharpened

      # Apply Unsharp Masking to the uploaded image
      sharpened_image = unsharp_mask(image)
      plt.imshow(cv2.cvtColor(sharpened_image, cv2.COLOR_BGR2RGB))
      plt.title("Unsharp Mask (Sharpening)")
      plt.axis('off')
      plt.show()
```

Unsharp Mask (Sharpening)

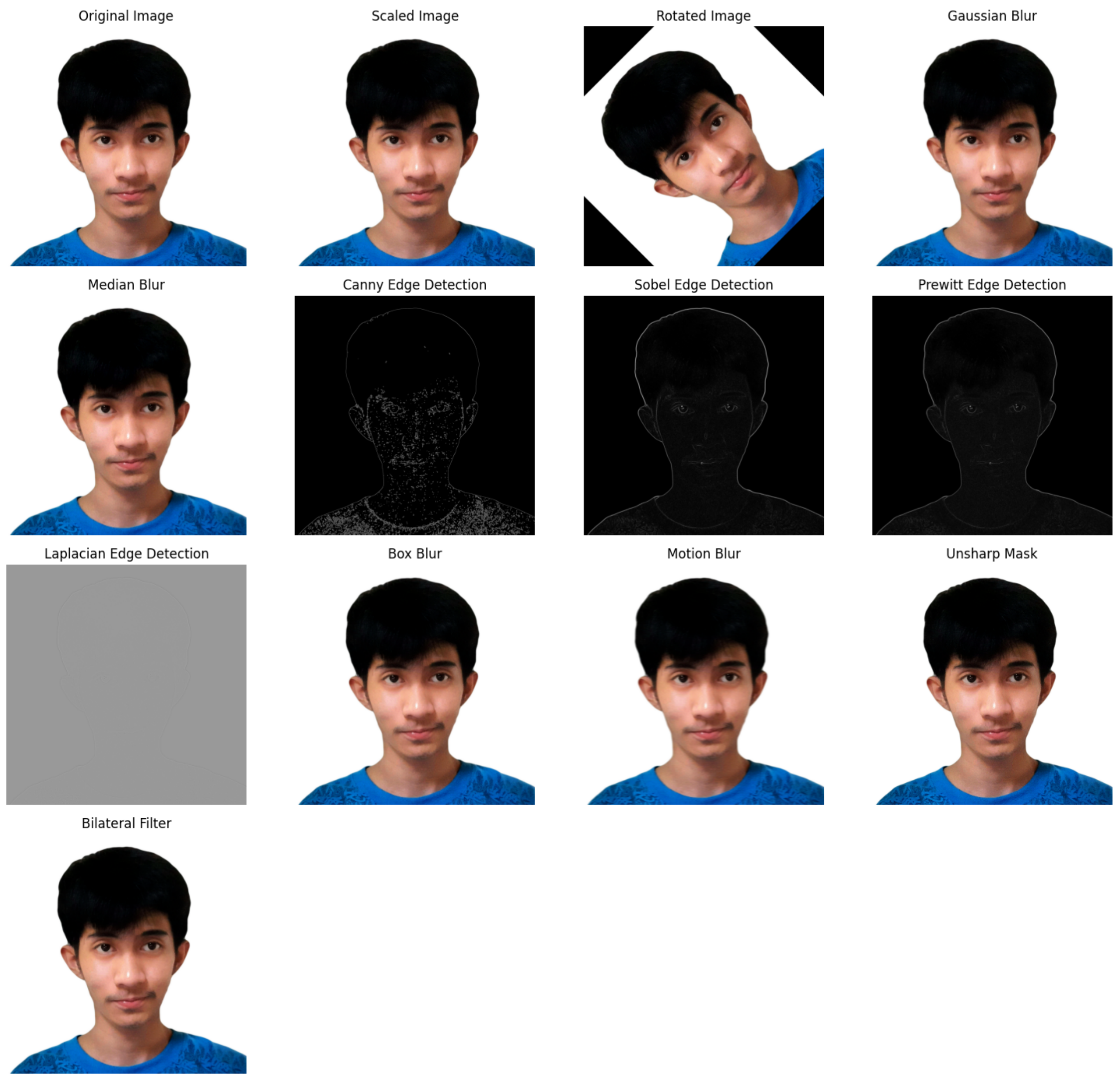
[16]:
```python
# Update process_image function to include new blurring techniques
def process_image(img, action):
  if action == 'scale':
    return scale_image(img, 0.5)
  elif action == 'rotate':
    return rotate_image(img, 45)
  elif action == 'gaussian_blur':
    return cv2.GaussianBlur(img, (5, 5), 0)
  elif action == 'median_blur':
    return cv2.medianBlur(img, 5)
  elif action == 'canny':
    return cv2.Canny(img, 100, 200)
  elif action == 'sobel':
    return sobel_edge_detection(img)
  elif action == 'laplacian':
    return laplacian_edge_detection(img)
  elif action == 'prewitt':
    return prewitt_edge_detection(img)
  elif action == 'bilateral_blur':
    return bilateral_blur(img)
  elif action == 'box_blur':
    return box_blur(img)
  elif action == 'motion_blur':
    return motion_blur(img)
  elif action == 'unsharp_mask':
    return unsharp_mask(img)
  else:
    return img

# Add new blurring options for interactive processing
# Add new blurring options for interactive processing
action = input("Enter action (scale, rotate, gaussian_blur, median_blur, canny,␣
 ↪sobel, laplacian, prewitt, bilateral_blur, box_blur, motion_b␣
 ↪processed_image = process_image(image, action)")
display_images(image, processed_image, "Original Image", f"Processed Image␣
 ↪({action})")
```

```
Enter action (scale, rotate, gaussian_blur, median_blur, canny, sobel,
laplacian, prewitt, bilateral_blur, box_blur, motion_b processed_image =
process_image(image, action)prewitt
```



[20]:
```
# Set up a large figure for the subplots
plt.figure(figsize=(15, 14))

# Display the original image
plt.subplot(4, 4, 1)
plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
plt.title("Original Image")
plt.axis('off')  # Hide the axis

# Display the scaled image
plt.subplot(4, 4, 2)
plt.imshow(cv2.cvtColor(scaled_image, cv2.COLOR_BGR2RGB))
plt.title("Scaled Image")
plt.axis('off')

# Display the rotated image
plt.subplot(4, 4, 3)
plt.imshow(cv2.cvtColor(rotated_image, cv2.COLOR_BGR2RGB))
plt.title("Rotated Image")
plt.axis('off')

# Display the image with Gaussian blur applied
plt.subplot(4, 4, 4)
plt.imshow(cv2.cvtColor(gaussian_blur, cv2.COLOR_BGR2RGB))
plt.title("Gaussian Blur")
plt.axis('off')

# Display the image with median blur applied
plt.subplot(4, 4, 5)
plt.imshow(cv2.cvtColor(median_blur, cv2.COLOR_BGR2RGB))
plt.title("Median Blur")
plt.axis('off')

# Display the image with Canny edge detection applied
plt.subplot(4, 4, 6)
plt.imshow(edges, cmap='gray')
plt.title("Canny Edge Detection")
plt.axis('off')

# Display the image with Sobel edge detection applied
plt.subplot(4, 4, 7)
plt.imshow(sobel_edges, cmap='gray')
plt.title("Sobel Edge Detection")
plt.axis('off')

# Display the image with Prewitt edge detection applied
plt.subplot(4, 4, 8)
plt.imshow(prewitt_edges, cmap='gray')
plt.title("Prewitt Edge Detection")
plt.axis('off')

# Display the image with Laplacian edge detection applied
plt.subplot(4, 4, 9)
plt.imshow(laplacian_edges, cmap='gray')
plt.title("Laplacian Edge Detection")
plt.axis('off')

# Display the image with Bilateral filter applied
```

```
plt.subplot(4, 4, 13)
plt.imshow(cv2.cvtColor(bilateral_filter, cv2.COLOR_BGR2RGB))
plt.title("Bilateral Filter")
plt.axis('off')

# Display the image with Box blur applied
plt.subplot(4, 4, 10)
plt.imshow(cv2.cvtColor(box_blurred, cv2.COLOR_BGR2RGB))
plt.title("Box Blur")
plt.axis('off')

# Display the image with motion blur applied
plt.subplot(4, 4, 11)
plt.imshow(cv2.cvtColor(motion_blurred, cv2.COLOR_BGR2RGB))
plt.title("Motion Blur")
plt.axis('off')

# Display the image with unsharp mask (sharpening) applied
plt.subplot(4, 4, 12)
plt.imshow(cv2.cvtColor(sharpened_image, cv2.COLOR_BGR2RGB))
plt.title("Unsharp Mask")
plt.axis('off')

# Adjust the layout of the subplots to avoid overlap and make everything fit
plt.tight_layout()
# Show the plot with all the images
plt.show()
```



14 Conclusion

You can implement the above code in Google Colab to carry out scaling, rotation, blurring, and edge detection tasks using OpenCV. These exercises will help your students understand how different image processing techniques work in practice.

[21]:
```python
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

[ ]:
```
!apt-get install -y pandoc
```

```
Reading package lists… Done
```

```
Building dependency tree… Done
Reading state information… Done
The following additional packages will be installed:
  libcmark-gfm-extensions0.29.0.gfm.3 libcmark-gfm0.29.0.gfm.3 pandoc-data
Suggested packages:
  texlive-luatex pandoc-citeproc context wkhtmltopdf librsvg2-bin groff ghc
nodejs php python
  libjs-mathjax libjs-katex citation-style-language-styles
The following NEW packages will be installed:
  libcmark-gfm-extensions0.29.0.gfm.3 libcmark-gfm0.29.0.gfm.3 pandoc pandoc-
data
0 upgraded, 4 newly installed, 0 to remove and 49 not upgraded.
Need to get 20.6 MB of archives.
After this operation, 156 MB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu jammy/universe amd64 libcmark-
gfm0.29.0.gfm.3 amd64 0.29.0.gfm.3-3 [115 kB]
Get:2 http://archive.ubuntu.com/ubuntu jammy/universe amd64 libcmark-gfm-
extensions0.29.0.gfm.3 amd64 0.29.0.gfm.3-3 [25.1 kB]
Get:3 http://archive.ubuntu.com/ubuntu jammy/universe amd64 pandoc-data all
2.9.2.1-3ubuntu2 [81.8 kB]
Get:4 http://archive.ubuntu.com/ubuntu jammy/universe amd64 pandoc amd64
2.9.2.1-3ubuntu2 [20.3 MB]
Fetched 20.6 MB in 1s (29.0 MB/s)
Selecting previously unselected package libcmark-gfm0.29.0.gfm.3:amd64.
```

[25]:
```
!jupyter nbconvert --to pdf "/content/drive/MyDrive/Colab Notebooks/Exercise 1␣
↪- Image Processing Techniques.ipynb"
```

```
[NbConvertApp] Converting notebook /content/drive/MyDrive/Colab
Notebooks/Exercise 1 - Image Processing Techniques.ipynb to pdf
[NbConvertApp] ERROR | Error while converting '/content/drive/MyDrive/Colab
Notebooks/Exercise 1 - Image Processing Techniques.ipynb'
Traceback (most recent call last):
  File "/usr/local/lib/python3.10/dist-packages/nbconvert/nbconvertapp.py", line
488, in export_single_notebook
    output, resources = self.exporter.from_filename(
  File "/usr/local/lib/python3.10/dist-
packages/nbconvert/exporters/exporter.py", line 189, in from_filename
    return self.from_file(f, resources=resources, **kw)
  File "/usr/local/lib/python3.10/dist-
packages/nbconvert/exporters/exporter.py", line 206, in from_file
    return self.from_notebook_node(
  File "/usr/local/lib/python3.10/dist-packages/nbconvert/exporters/pdf.py",
line 181, in from_notebook_node
    latex, resources = super().from_notebook_node(nb, resources=resources, **kw)
  File "/usr/local/lib/python3.10/dist-packages/nbconvert/exporters/latex.py",
line 74, in from_notebook_node
    return super().from_notebook_node(nb, resources, **kw)
  File "/usr/local/lib/python3.10/dist-
packages/nbconvert/exporters/templateexporter.py", line 413, in
from_notebook_node
    output = self.template.render(nb=nb_copy, resources=resources)
  File "/usr/local/lib/python3.10/dist-packages/jinja2/environment.py", line
1304, in render
    self.environment.handle_exception()
  File "/usr/local/lib/python3.10/dist-packages/jinja2/environment.py", line
939, in handle_exception
    raise rewrite_traceback_stack(source=source)
  File "/usr/local/share/jupyter/nbconvert/templates/latex/index.tex.j2", line
8, in top-level template code
    ((* extends cell_style *))
  File
"/usr/local/share/jupyter/nbconvert/templates/latex/style_jupyter.tex.j2", line
176, in top-level template code
    \prompt{(((prompt)))}{(((prompt_color)))}{(((execution_count)))}{(((extra_sp
ace)))}
  File "/usr/local/share/jupyter/nbconvert/templates/latex/base.tex.j2", line 7,
in top-level template code
    ((*- extends 'document_contents.tex.j2' -*))
  File
"/usr/local/share/jupyter/nbconvert/templates/latex/document_contents.tex.j2",
line 51, in top-level template code
    ((*- block figure scoped -*))
  File "/usr/local/share/jupyter/nbconvert/templates/latex/display_priority.j2",
line 5, in top-level template code
    ((*- extends 'null.j2' -*))
  File "/usr/local/share/jupyter/nbconvert/templates/latex/null.j2", line 30, in
top-level template code
    ((*- block body -*))
  File "/usr/local/share/jupyter/nbconvert/templates/latex/base.tex.j2", line
215, in block 'body'
    ((( super() )))
  File "/usr/local/share/jupyter/nbconvert/templates/latex/null.j2", line 32, in
block 'body'
    ((*- block any_cell scoped -*))
  File "/usr/local/share/jupyter/nbconvert/templates/latex/null.j2", line 85, in
block 'any_cell'
    ((*- block markdowncell scoped-*)) ((*- endblock markdowncell -*))
  File
"/usr/local/share/jupyter/nbconvert/templates/latex/document_contents.tex.j2",
line 68, in block 'markdowncell'
    ((( cell.source | citation2latex | strip_files_prefix |
convert_pandoc('markdown+tex_math_double_backslash', 'json',extra_args=[]) |
resolve_references | convert_pandoc('json','latex'))))
  File "/usr/local/lib/python3.10/dist-packages/nbconvert/filters/pandoc.py",
line 24, in convert_pandoc
```

```
    return pandoc(source, from_format, to_format, extra_args=extra_args)
  File "/usr/local/lib/python3.10/dist-packages/nbconvert/utils/pandoc.py", line
51, in pandoc
    check_pandoc_version()
  File "/usr/local/lib/python3.10/dist-packages/nbconvert/utils/pandoc.py", line
99, in check_pandoc_version
    v = get_pandoc_version()
  File "/usr/local/lib/python3.10/dist-packages/nbconvert/utils/pandoc.py", line
76, in get_pandoc_version
    raise PandocMissing()
nbconvert.utils.pandoc.PandocMissing: Pandoc wasn't found.
Please check that pandoc is installed:
https://pandoc.org/installing.html
```