LAB_1.py

```python
import numpy as np
import matplotlib.pyplot as plt
from skimage import io, color, util, img_as_float
from scipy.ndimage import convolve
from scipy.ndimage import gaussian_filter
from skimage.filters import gaussian
from skimage.restoration import denoise_bilateral
from skimage.filters.rank import median
from skimage.morphology import disk
from skimage.util import random_noise
from skimage.filters import rank

img = img_as_float(color.rgb2gray(io.imread(r'Low Contrast.jpg')))

gauss_noise = random_noise(img, mode='gaussian', var=0.01)
sp_noise = random_noise(img, mode='s&p', amount=0.05)

avg_kernel = np.ones((3, 3)) / 9
gauss_sigma = 0.5

results = [
    (img, 'Original'),
    (gauss_noise, 'Gaussian Noise'),
    (sp_noise, 'Salt & Pepper'),
    (convolve(gauss_noise, avg_kernel), 'Mean (Gauss)'),
    (convolve(sp_noise, avg_kernel), 'Mean (S&P)'),
    (median((gauss_noise * 255).astype(np.uint8), disk(1)) / 255.0, 'Median (Gauss)'),
    (median((sp_noise * 255).astype(np.uint8), disk(1)) / 255.0, 'Median (S&P)'),
    (gaussian(gauss_noise, sigma=gauss_sigma), 'Gauss (Gauss)'),
    (gaussian(sp_noise, sigma=gauss_sigma), 'Gauss (S&P)')
]

fig, axes = plt.subplots(3, 3, figsize=(12, 12))
for ax, (image, title) in zip(axes.ravel(), results):
    ax.imshow(image, cmap='gray')
    ax.set_title(title)
    ax.axis('off')
plt.tight_layout()
plt.show()

```

LAB_2.py

```python
import numpy as np
import matplotlib.pyplot as plt
from skimage import io, color, img_as_float, exposure

image = io.imread(r'Low Contrast.jpg')
gray_image = color.rgb2gray(image)
gray_image = img_as_float(gray_image)

global_he = exposure.equalize_hist(gray_image)
clahe = exposure.equalize_adapthist(gray_image, clip_limit=0.02, nbins=256)

plt.figure(figsize=(12, 8))
plt.subplot(3, 3, 1)
plt.imshow(gray_image, cmap='gray')
plt.title('Original')
plt.axis('off')

plt.subplot(3, 3, 2)
plt.imshow(global_he, cmap='gray')
plt.title('Global HE')
plt.axis('off')

plt.subplot(3, 3, 3)
plt.imshow(clahe, cmap='gray')
plt.title('CLAHE')
plt.axis('off')

plt.subplot(3, 3, 4)
plt.hist(gray_image.ravel(), bins=256, range=(0, 1), color='black')
plt.title('Original Histogram')

plt.subplot(3, 3, 5)
plt.hist(global_he.ravel(), bins=256, range=(0, 1), color='black')
plt.title('Global HE Histogram')

plt.subplot(3, 3, 6)
plt.hist(clahe.ravel(), bins=256, range=(0, 1), color='black')
plt.title('CLAHE Histogram')

# Step 4: Print contrast (standard deviation)
print("Contrast (Standard Deviation):")
print(f"Original:   {np.std(gray_image):.4f}")
print(f"Global HE:  {np.std(global_he):.4f}")
print(f"CLAHE:      {np.std(clahe):.4f}")

plt.tight_layout()
plt.show()
```

Content Script

LAB_3.py

```python
import matplotlib.pyplot as plt
from skimage import io, color, filters
from skimage.filters import threshold_otsu
from skimage.filters import threshold_local

image = io.imread(r'Low Contrast.jpg')
if image.ndim == 3:
    gray_image = color.rgb2gray(image)
else:
    gray_image = image

otsu_thresh = threshold_otsu(gray_image)
otsu_binary = gray_image > otsu_thresh

adaptive_thresh = threshold_local(gray_image, block_size=35, offset=0.0)
adaptive_binary = gray_image > adaptive_thresh

plt.figure(figsize=(12, 4))

plt.subplot(1, 3, 1)
plt.imshow(gray_image, cmap='gray')
plt.title('Original')
plt.axis('off')

plt.subplot(1, 3, 2)
plt.imshow(otsu_binary, cmap='gray')
plt.title("Otsu's Threshold")
plt.axis('off')

plt.subplot(1, 3, 3)
plt.imshow(adaptive_binary, cmap='gray')
plt.title('Adaptive Threshold')
plt.axis('off')

plt.tight_layout()
plt.show()

```

Content Script

LAB_4.py

```python
import matplotlib.pyplot as plt
import numpy as np
from skimage import io, color, filters, feature, img_as_float

image = io.imread(r'Low Contrast.jpg')
gray_image = color.rgb2gray(image)
gray_image = img_as_float(gray_image)

sobel_edges = filters.sobel(gray_image)
prewitt_edges = filters.prewitt(gray_image)
canny_edges = feature.canny(gray_image)

plt.figure(figsize=(10, 8))

plt.subplot(2, 2, 1)
plt.imshow(gray_image, cmap='gray')
plt.title('Original')
plt.axis('off')

plt.subplot(2, 2, 2)
plt.imshow(sobel_edges, cmap='gray')
plt.title('Sobel')
plt.axis('off')

plt.subplot(2, 2, 3)
plt.imshow(prewitt_edges, cmap='gray')
plt.title('Prewitt')
plt.axis('off')

plt.subplot(2, 2, 4)
plt.imshow(canny_edges, cmap='gray')
plt.title('Canny')
plt.axis('off')

plt.tight_layout()
plt.show()

fig, ax = plt.subplots(1, 3, figsize=(12, 4))
titles = ['Sobel', 'Prewitt', 'Canny']
edges = [sobel_edges, prewitt_edges, canny_edges]

for i in range(3):
    ax[i].imshow(edges[i], cmap='gray')
    ax[i].set_title(titles[i])
    ax[i].axis('off')
plt.suptitle('Sobel | Prewitt | Canny', fontsize=14)
plt.tight_layout()
plt.show()
```

Content Script

```
49
50 sobel_density = 100 * np.count_nonzero(sobel_edges > 0) / sobel_edges.size
51 prewitt_density = 100 * np.count_nonzero(prewitt_edges > 0) / prewitt_edges.size
52 canny_density = 100 * np.count_nonzero(canny_edges) / canny_edges.size
53
54 print("Edge Density:")
55 print(f"Sobel:   {sobel_density:.2f}%")
56 print(f"Prewitt: {prewitt_density:.2f}%")
57 print(f"Canny:   {canny_density:.2f}%")
58
```

Content Script

LAB_5.py

```python
import matplotlib.pyplot as plt
import numpy as np
from skimage import io, color, filters, morphology, measure
from skimage.filters import threshold_local
from skimage.util import invert

image = io.imread(r'Low Contrast.jpg')
gray = color.rgb2gray(image)

adaptive_thresh = threshold_local(gray, block_size=35, offset=0.1)
binary = gray < adaptive_thresh

cleaned = morphology.remove_small_objects(binary, min_size=50)

label_image = measure.label(cleaned)
regions = measure.regionprops(label_image)

regions = sorted(regions, key=lambda r: r.bbox[1])

plt.figure(figsize=(10, 8))
plt.subplot(2, 2, 1)
plt.imshow(image)
plt.title('Original Image')
plt.axis('off')

plt.subplot(2, 2, 2)
plt.imshow(~binary, cmap='gray')
plt.title('Binarized Image')
plt.axis('off')

plt.subplot(2, 2, 3)
plt.imshow(cleaned, cmap='gray')
plt.title('Cleaned Image')
plt.axis('off')

plt.subplot(2, 2, 4)
plt.imshow(image)
plt.title('Detected Characters')
plt.axis('off')
for region in regions:
    minr, minc, maxr, maxc = region.bbox
    rect = plt.Rectangle((minc, minr), maxc - minc, maxr - minr,
                         edgecolor='red', facecolor='none', linewidth=1)
    plt.gca().add_patch(rect)
plt.tight_layout()
plt.show()

num_chars = len(regions)
```

Content Script

```
cols = 5
rows = int(np.ceil(num_chars / cols))

plt.figure(figsize=(15, rows * 2))
for i, region in enumerate(regions):
    plt.subplot(rows, cols, i + 1)
    plt.imshow(region.image, cmap='gray')
    plt.title(f'{i+1}')
    plt.axis('off')
plt.suptitle('Segmented Characters')
plt.tight_layout()
plt.show()

print(f"Total characters detected: {num_chars}")

```

Content Script

LAB_6.py

```python
import matplotlib.pyplot as plt
import numpy as np
from skimage import io, color, filters, measure, morphology
from skimage.filters import threshold_local

img = io.imread(r'rice (1).tiff')
gray = color.rgb2gray(img)

block_size = 35
adaptive_thresh = threshold_local(gray, block_size, offset=10)
binary = gray < adaptive_thresh
cleaned = morphology.remove_small_objects(binary, min_size=50)

label_img = measure.label(cleaned)
regions = measure.regionprops(label_img)

plt.figure(figsize=(8, 8))
plt.imshow(gray, cmap='gray')
plt.title('Rice Grains')
for i, region in enumerate(regions):
    y, x = region.centroid
    plt.text(x, y, str(i + 1), color='red', fontsize=8)
plt.axis('off')
plt.show()

areas = np.array([r.area for r in regions])
majors = np.array([r.major_axis_length for r in regions])
perims = np.array([r.perimeter for r in regions])

print(f"Total grains: {len(regions)}\n")
print(f"{'No.':<5} {'Area':<7} {'MajorAxisLen':<15} {'Perimeter':<10}")
for i in range(len(regions)):
    print(f"{i+1:<5} {areas[i]:<7.1f} {majors[i]:<15.1f} {perims[i]:<10.1f}")

print(f"\nMin: {areas.min():.1f}, Max: {areas.max():.1f}, Mean: {areas.mean():.1f}")
in_range = np.sum((areas >= 200) & (areas <= 400))
print(f"Grains in area [200–400]: {in_range}")

```

Content Script

LAB_7.py

```python
import matplotlib.pyplot as plt
import numpy as np
from skimage import io, color, img_as_float
from scipy.signal import convolve2d

img = io.imread(r'Low Contrast.jpg')
gray = color.rgb2gray(img)
gray = img_as_float(gray)

mask = np.ones((3, 3)) / 9

print("Convolution Mask:")
print(mask)

conv_img = convolve2d(gray, mask, mode='same', boundary='symm')

plt.figure(figsize=(10, 4))

plt.subplot(1, 2, 1)
plt.imshow(gray, cmap='gray')
plt.title('Original')
plt.axis('off')

plt.subplot(1, 2, 2)
plt.imshow(conv_img, cmap='gray')
plt.title('Convolved')
plt.axis('off')

plt.tight_layout()
plt.show()

```

Content Script