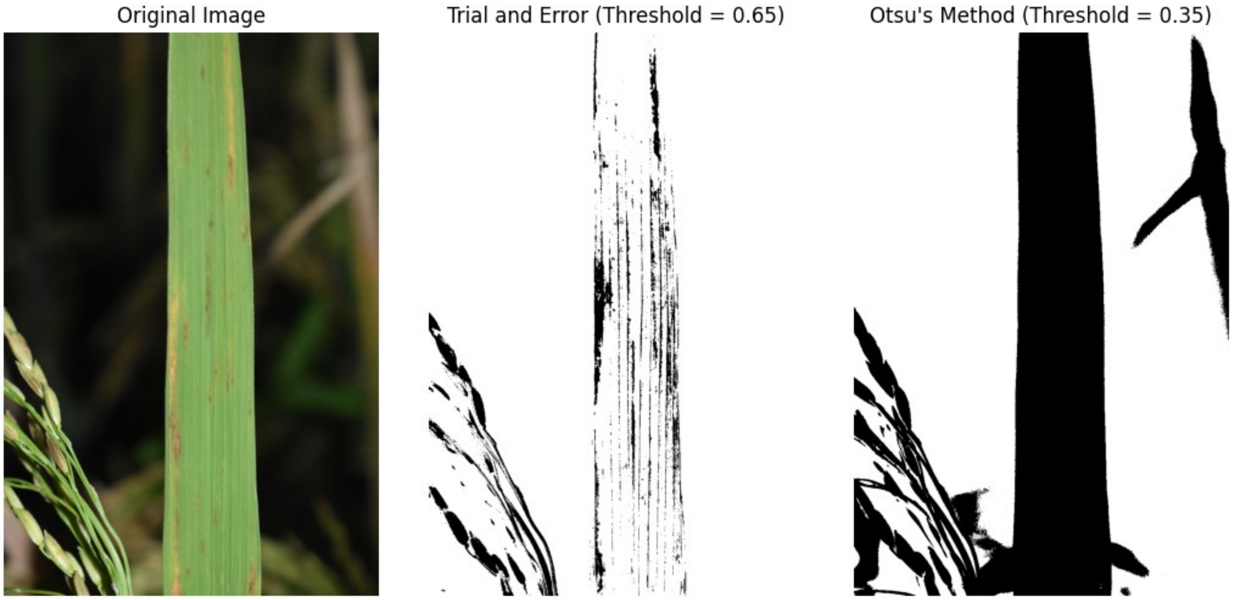
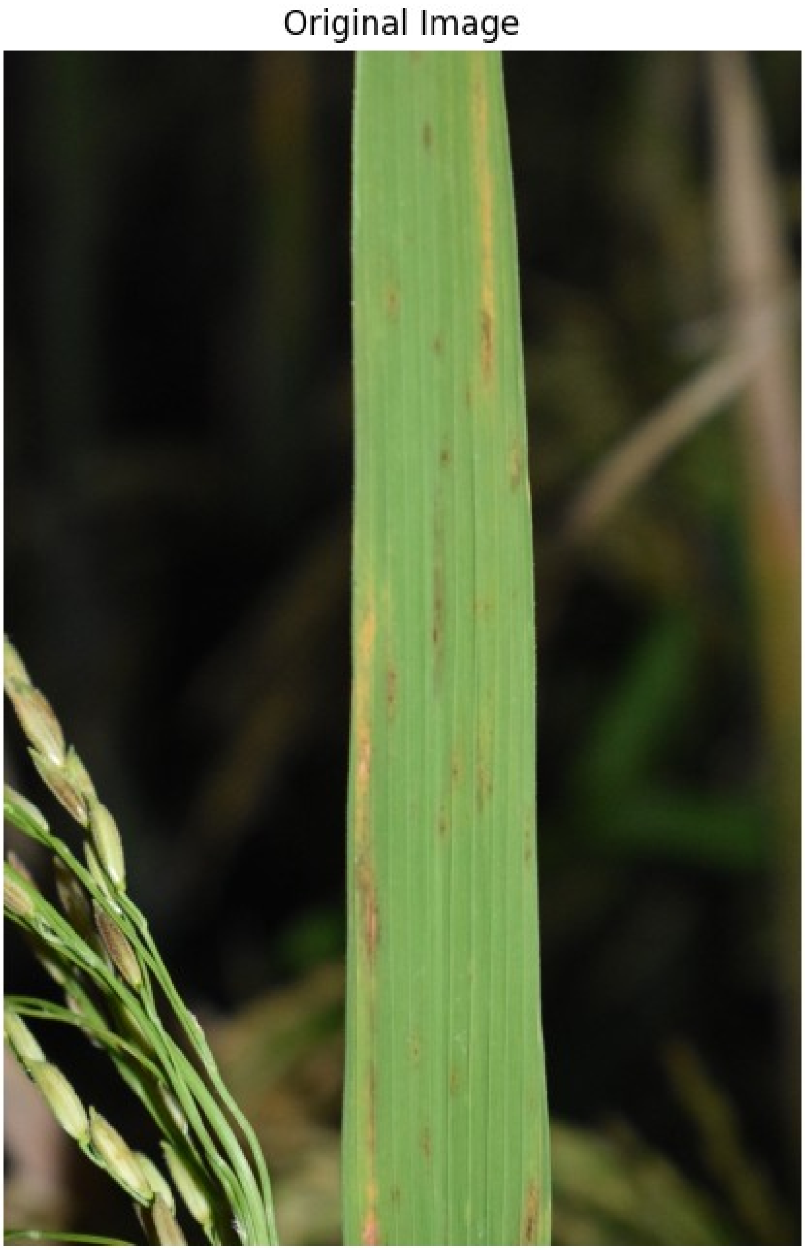
```python
import numpy as np
import matplotlib.pyplot as plt
from skimage.io import imread, imshow
from skimage.color import rgb2gray, rgb2hsv
from skimage.filters import threshold_otsu

from google.colab import drive
drive.mount('/content/drive')

# Load Original Image
img_test1_path = '/content/drive/My Drive/Colab Notebooks/image1.jpg'
img_test1 = imread(img_test1_path)

# Convert to Grayscale
img_test1_gs = rgb2gray(img_test1)

# After several trial and error this is the best threshold
th = 0.65
img_test1_bn = img_test1_gs < th

# Using Otsu's Method
th_otsu = threshold_otsu(img_test1_gs)
img_test1_otsu = img_test1_gs < th_otsu

# Plot
fig, ax = plt.subplots(1, 3, figsize=(18, 6))
fig.subplots_adjust(wspace=-0.5)
ax[0].set_title("Original Image")
ax[0].imshow(img_test1, cmap='gray')
ax[0].set_axis_off()
ax[1].set_title(f"Trial and Error (Threshold = {th:.2f})")
ax[1].imshow(img_test1_bn, cmap='gray')
ax[1].set_axis_off()
ax[2].set_title(f"Otsu's Method (Threshold = {th_otsu:.2f})")
ax[2].imshow(img_test1_otsu, cmap='gray')
ax[2].set_axis_off()

plt.show()
```

```
Drive already mounted at /content/drive; to attempt to forcibly
remount, call drive.mount("/content/drive", force_remount=True).
```


Original Image
Trial and Error (Threshold = 0.65)
Otsu's Method (Threshold = 0.35)

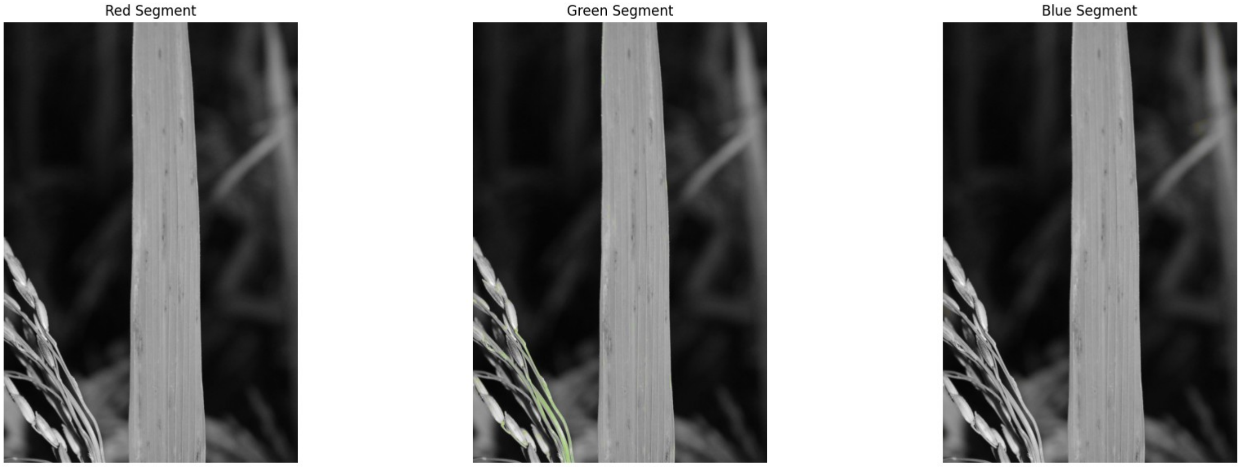
```
# Laod Image
img = imread('/content/drive/My Drive/Colab Notebooks/image1.jpg')
[:, :, :3]
img_gs_1c = rgb2gray(img)

# Plot
fig, ax = plt.subplots(1, 1, figsize=(8, 8))
ax.set_title("Original Image")
ax.imshow(img)
ax.set_axis_off()
plt.show()
```



Original Image

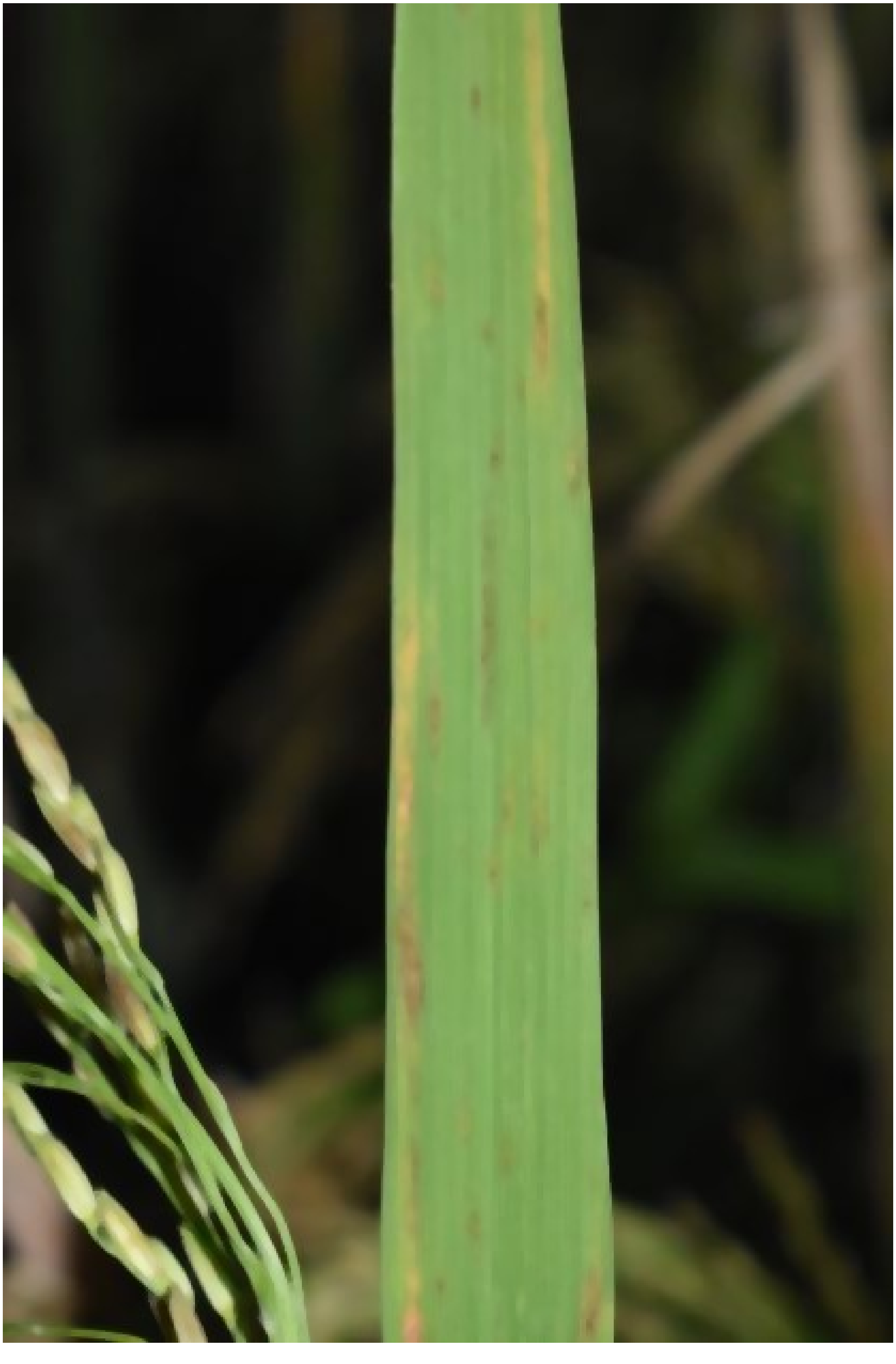
```python
# Grayscale image with 3 channels (the value is triplicated)
img_gs = ((np.stack([img_gs_1c] * 3, axis=-1) * 255)
          .astype('int').clip(0, 255))

# Red mask
red_mask = ((img[:, :, 0] > 150) &
            (img[:, :, 1] < 100) &
            (img[:, :, 2] < 200))
img_red = img_gs.copy()
img_red[red_mask] = img[red_mask]
```

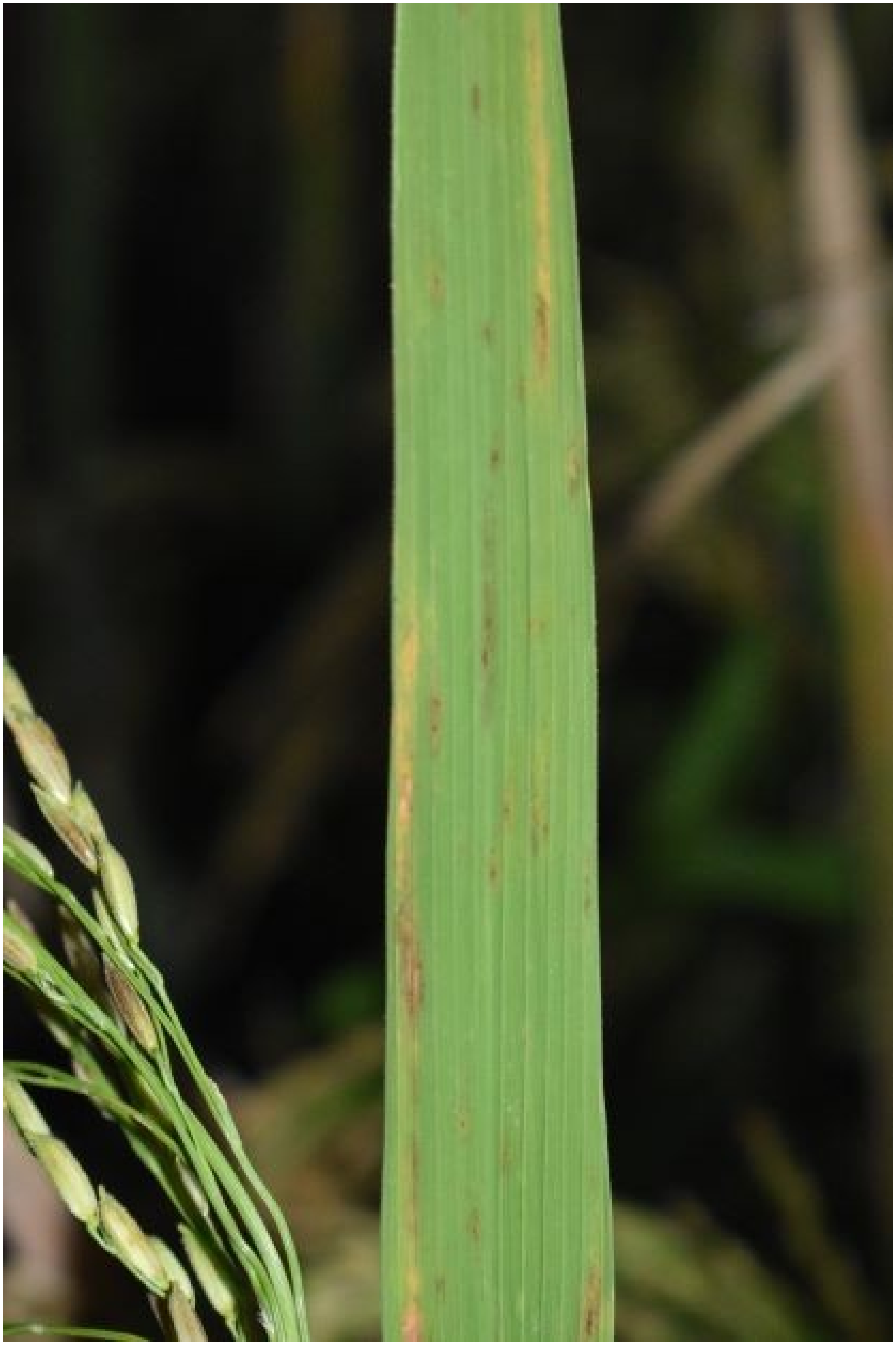
```
# Green mask
green_mask = ((img[:, :, 0] < 190) &
              (img[:, :, 1] > 190) &
              (img[:, :, 2] < 190))
img_green = img_gs.copy()
img_green[green_mask] = img[green_mask]

# Blue mask
blue_mask = ((img[:, :, 0] < 80) &
             (img[:, :, 1] < 85) &
             (img[:, :, 2] > 50))
img_blue = img_gs.copy()
img_blue[blue_mask] = img[blue_mask]

# Plot
fig, ax = plt.subplots(1, 3, figsize=(21, 7))
ax[0].set_title("Red Segment")
ax[0].imshow(img_red)
ax[0].set_axis_off()
ax[1].set_title("Green Segment")
ax[1].imshow(img_green)
ax[1].set_axis_off()
ax[2].set_title("Blue Segment")
ax[2].imshow(img_blue)
ax[2].set_axis_off()
plt.show()
```



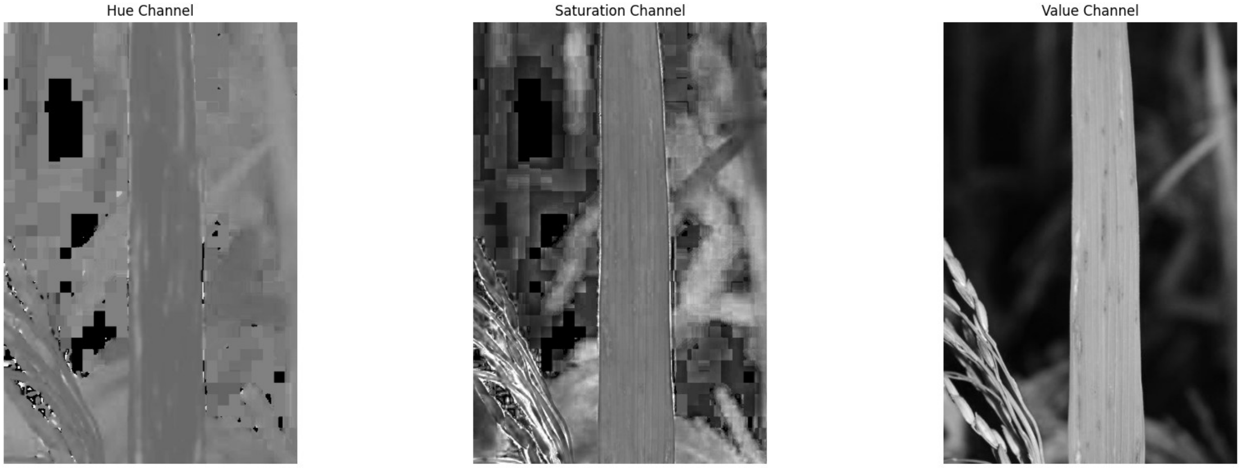
```
import cv2
import numpy as np
from google.colab.patches import cv2_imshow

# Reading an image from computer and taking dimensions
img = cv2.imread('/content/drive/My Drive/Colab Notebooks/image1.jpg')
rows, cols = img.shape[:2]
```

```
# Box Filter
output_box = cv2.boxFilter(img, -1, (5, 5), normalize=False)
cv2_imshow(output_box)

# Gaussian Blur
output_gaus = cv2.GaussianBlur(img, (5, 5), 0)
cv2_imshow(output_gaus)

# Median Blur (reduction of noise)
output_med = cv2.medianBlur(img, 5)
cv2_imshow(output_med)

# Bilateral filtering (Reduction of noise + Preserving of edges)
output_bil = cv2.bilateralFilter(img, 5, 6, 6)
cv2_imshow(output_bil)

# Original Image
cv2_imshow(img)
```

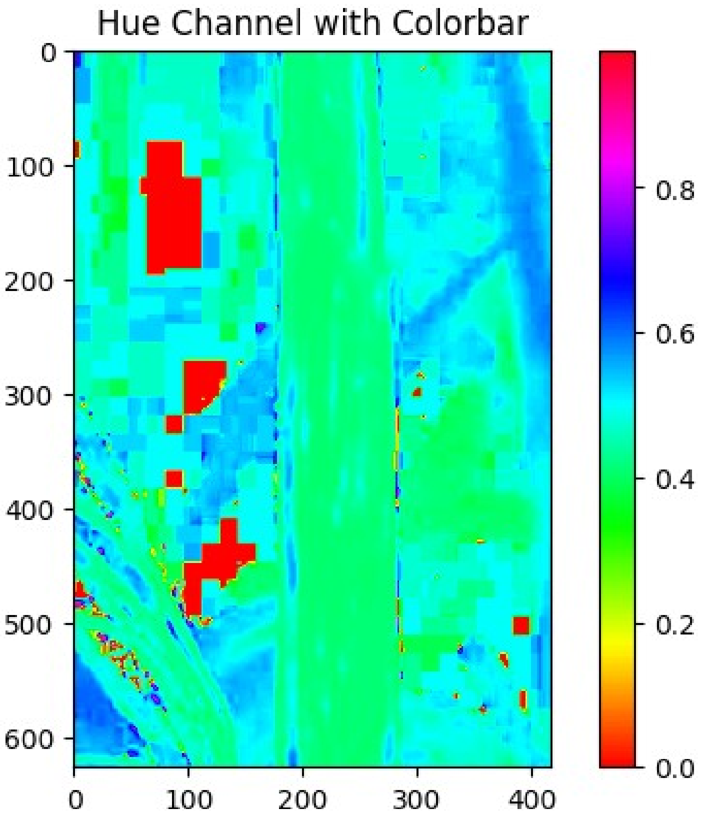




```
# Convert to HSV
img_hsv = rgb2hsv(img)

# Plot
fig, ax = plt.subplots(1, 3, figsize=(21, 7))
ax[0].set_title("Hue Channel")
ax[0].imshow(img_hsv[:, :, 0], cmap='gray')
ax[0].set_axis_off()
ax[1].set_title("Saturation Channel")
ax[1].imshow(img_hsv[:, :, 1], cmap='gray')
ax[1].set_axis_off()
ax[2].set_title("Value Channel")
ax[2].imshow(img_hsv[:, :, 2], cmap='gray')
ax[2].set_axis_off()
plt.show()
```

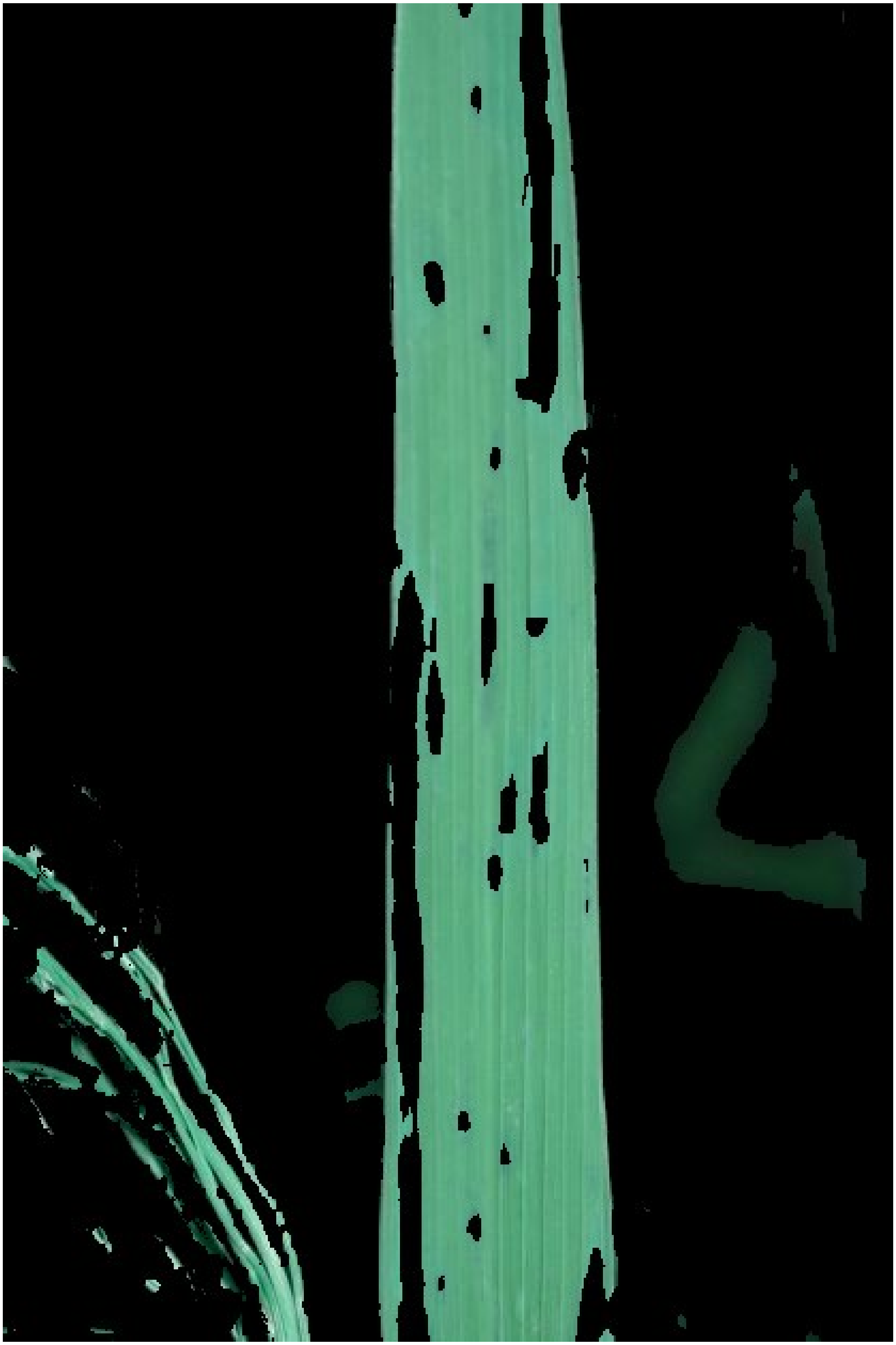


```
# Plot Hue Channel with Colorbar
plt.imshow(img_hsv[:, :, 0], cmap='hsv')
plt.title('Hue Channel with Colorbar')
plt.colorbar()
plt.show()
```

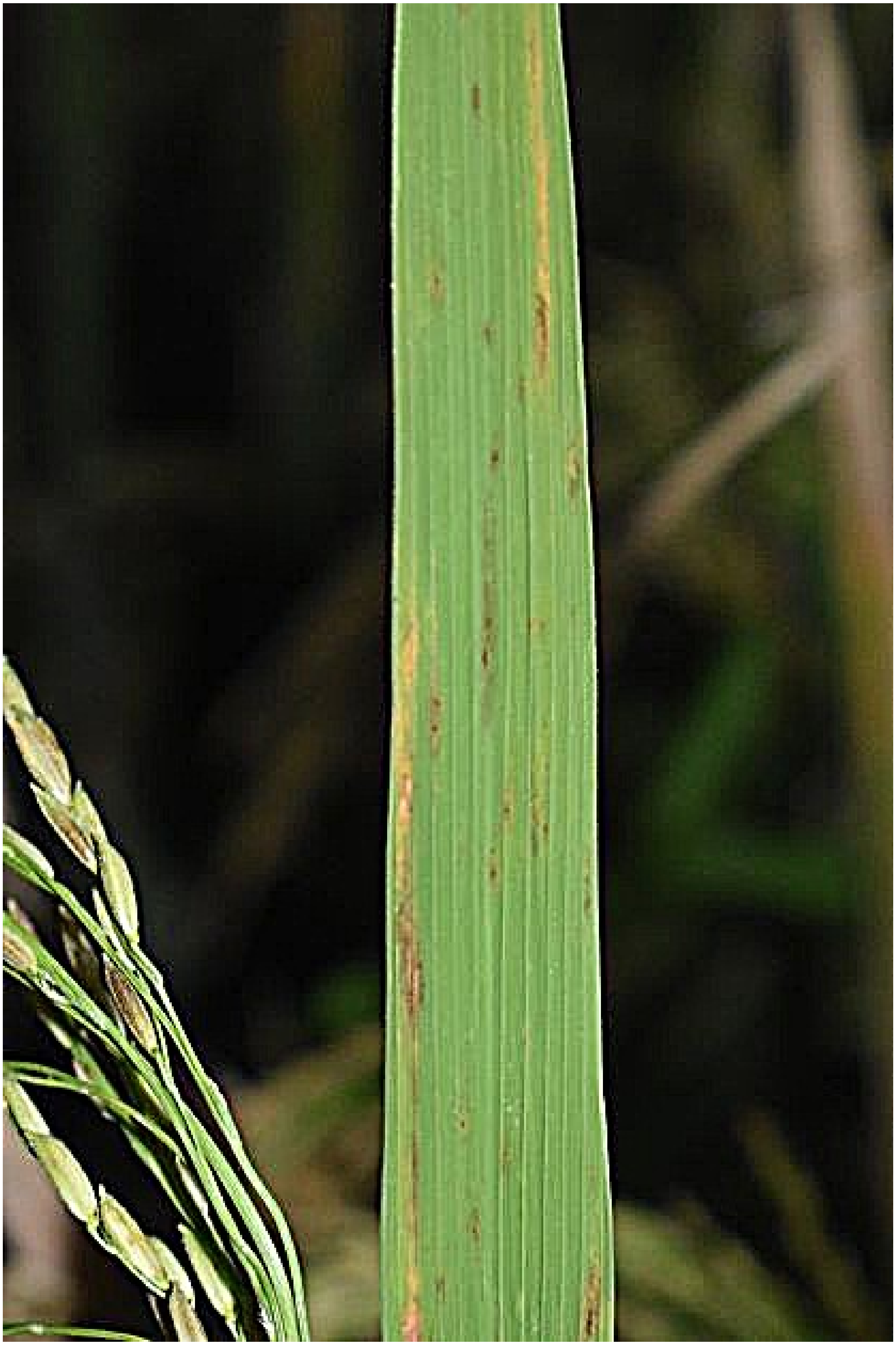


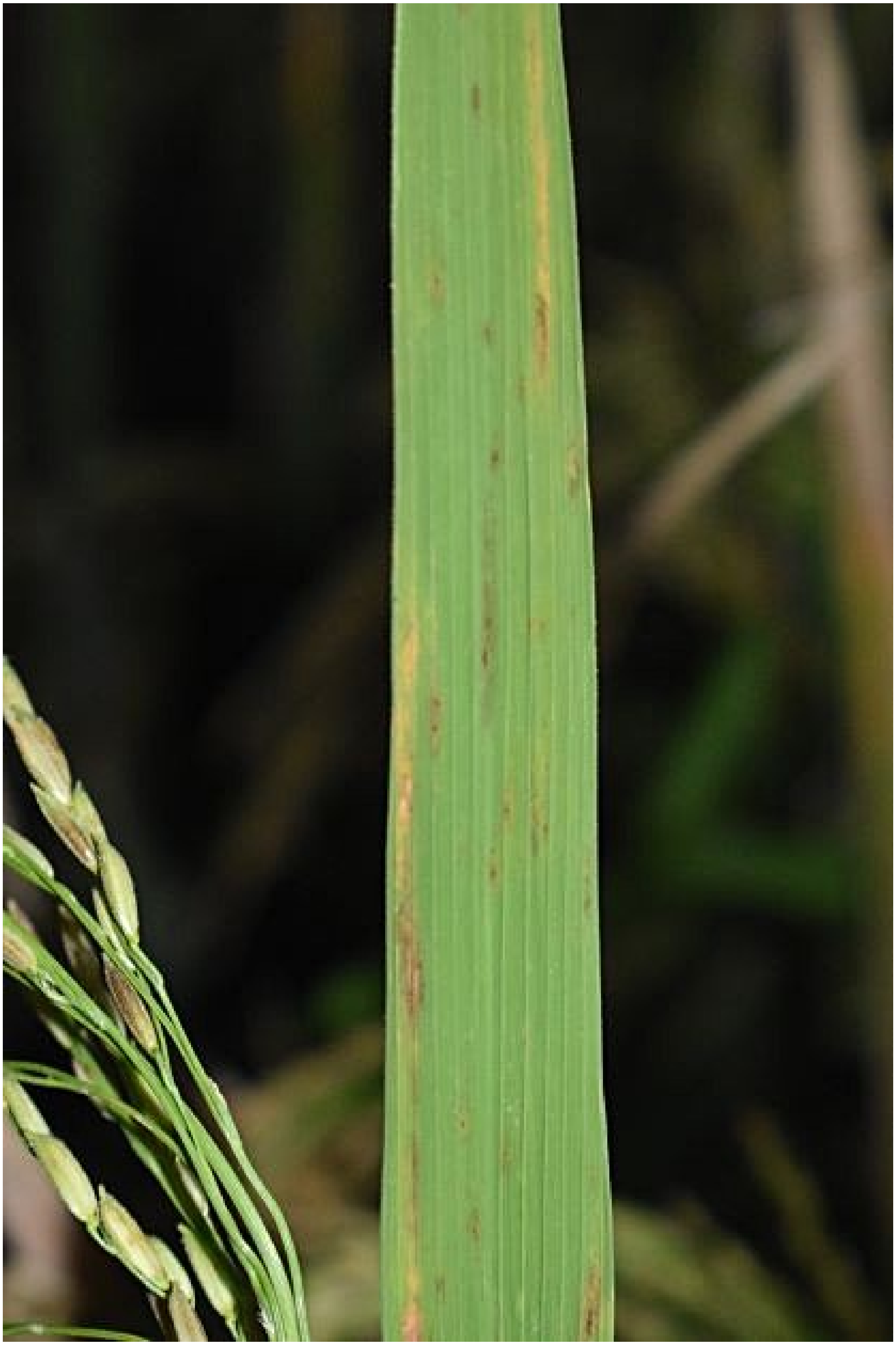
```python
import cv2
import numpy as np
from google.colab.patches import cv2_imshow

# Reading an image from computer
img = cv2.imread('/content/drive/My Drive/Colab Notebooks/image1.jpg')

# Convert the image to HSV color space
img_hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)

# Define the lower and upper bounds for the green color in HSV
lower_green = np.array([40, 40, 40])  # Example lower bound for green
upper_green = np.array([80, 255, 255]) # Example upper bound for green

# Create a mask using inRange function to identify green color
mask = cv2.inRange(img_hsv, lower_green, upper_green)

# Bitwise AND operation to apply the mask to the original image
output_masked = cv2.bitwise_and(img, img, mask=mask)

# Display the mask
cv2_imshow(mask)

# Display the original image and the masked image
```

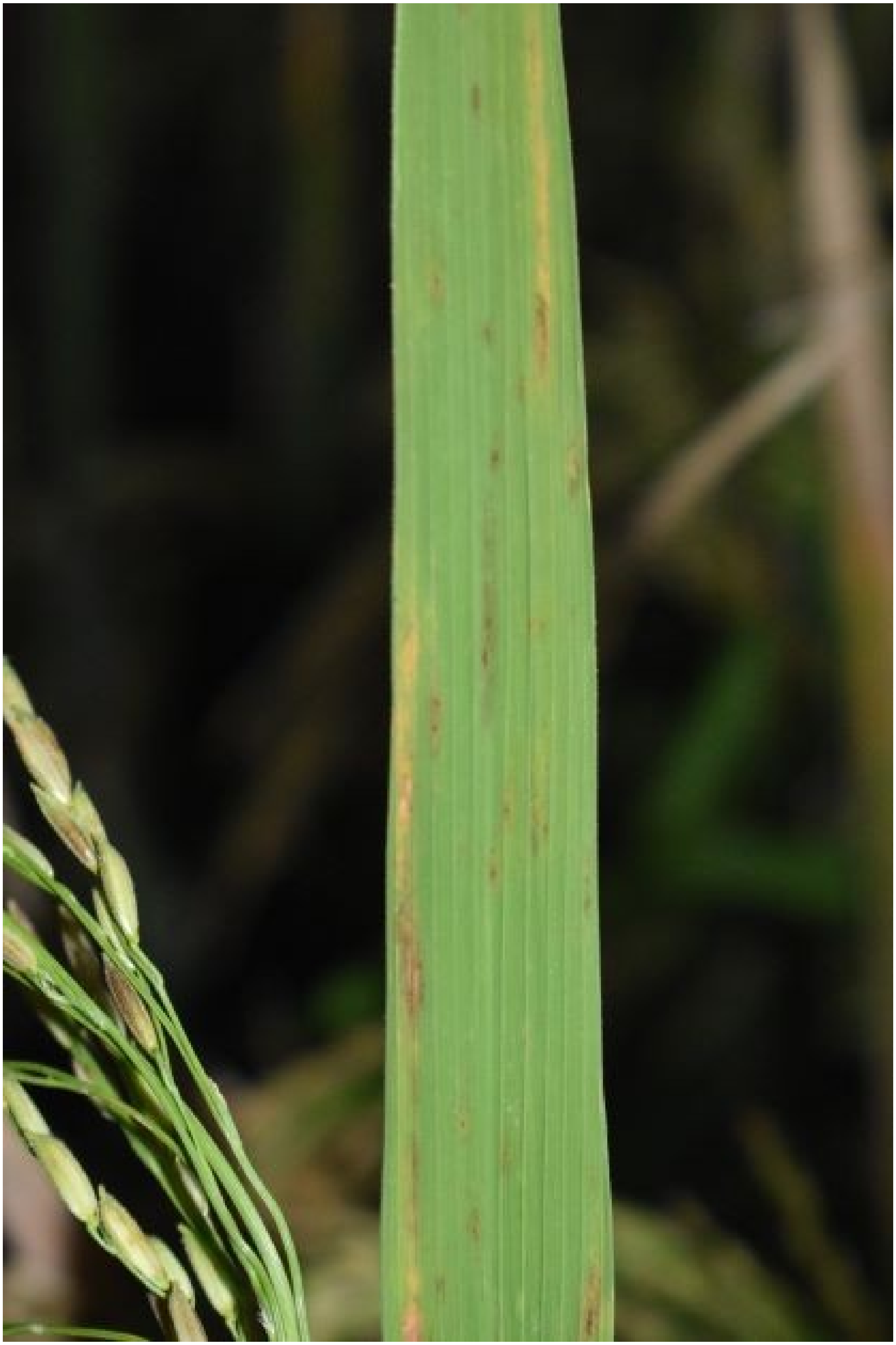
```
cv2_imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
cv2_imshow(cv2.cvtColor(output_masked, cv2.COLOR_BGR2RGB))
```



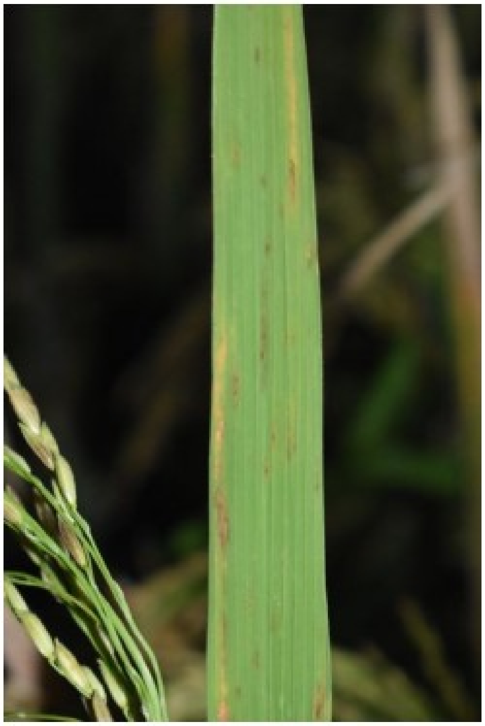
```
import cv2
import numpy as np
from google.colab.patches import cv2_imshow

# Reading the image
img = cv2.imread('/content/drive/My Drive/Colab
Notebooks/example1.jpg')

# Gaussian kernel for sharpening
gaussian_blur = cv2.GaussianBlur(img, (7,7), 2)

# Sharpening using addweighted()
sharpened1 = cv2.addWeighted(img, 1.5, gaussian_blur, -0.5, 0)
sharpened2 = cv2.addWeighted(img, 3.5, gaussian_blur, -2.5, 0)
sharpened3 = cv2.addWeighted(img, 7.5, gaussian_blur, -6.5, 0)

# Showing the sharpened Images one after the other
cv2_imshow(sharpened3)
cv2.waitKey(0)
cv2_imshow(sharpened2)
cv2.waitKey(0)
cv2_imshow(sharpened1)
cv2.waitKey(0)
cv2_imshow(img)
cv2.waitKey(0)

# Close all OpenCV windows
cv2.destroyAllWindows()
```

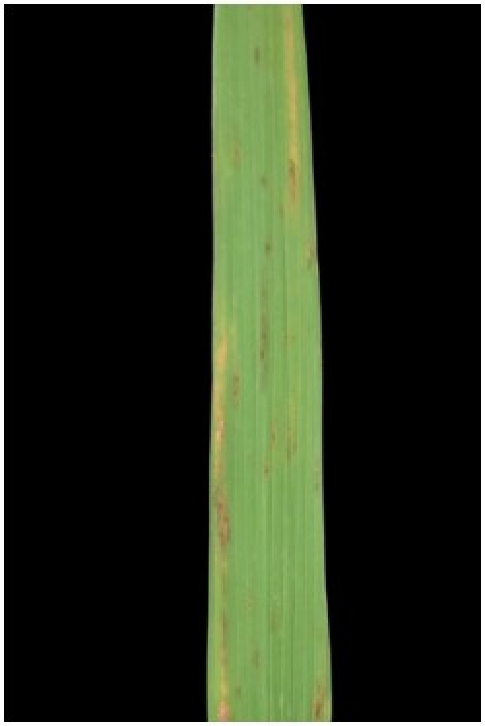






```
from rembg import remove
from PIL import Image
import matplotlib.pyplot as plt

input_path = '/content/drive/My Drive/Colab Notebooks/image1.jpg'
output_path = '/content/drive/My Drive/Colab Notebooks/image1_output.jpg'

# Open input image
input_image = Image.open(input_path)
# Display input image
plt.imshow(input_image)
plt.axis('off')  # Turn off axis
plt.show()

# Remove background
output_image = remove(input_image)

# Convert output image to RGB mode
output_image = output_image.convert("RGB")

# Display output image
plt.imshow(output_image)
plt.axis('off')  # Turn off axis
plt.show()

# Save output image
output_image.save(output_path)
```





```python
import cv2

# Load the original image
img = cv2.imread('/content/drive/My Drive/Colab Notebooks/image1.jpg')
```

```python
# Rotate the image
img_rotate_90_clockwise = cv2.rotate(img, cv2.ROTATE_90_CLOCKWISE)
img_rotate_90_counterclockwise = cv2.rotate(img,
cv2.ROTATE_90_COUNTERCLOCKWISE)
img_rotate_180 = cv2.rotate(img, cv2.ROTATE_180)

# Concatenate images horizontally
concatenated_img = cv2.hconcat([img, img_rotate_90_clockwise,
img_rotate_90_counterclockwise, img_rotate_180])

# Check if any of the images are empty
if img is None or img_rotate_90_clockwise is None or
img_rotate_90_counterclockwise is None or img_rotate_180 is None:
    print("Error: One or more images are empty or not loaded
properly.")
else:
    # Save the concatenated image
    cv2.imwrite('/content/drive/My Drive/Colab
Notebooks/newimage1.jpg', concatenated_img)
    print('Original image:', img.shape)
    print('Rotated 90 clockwise:', img_rotate_90_clockwise.shape)
    print('Rotated 90 counterclockwise:',
img_rotate_90_counterclockwise.shape)
    print('Rotated 180:', img_rotate_180.shape)
```

```
Error: One or more images are empty or not loaded properly.
```