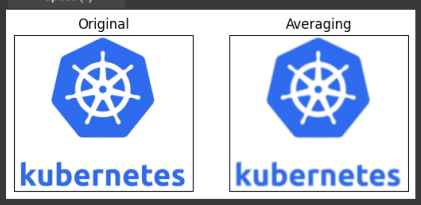
Filter Moving Average

```
[7] import numpy as np
    import cv2 as cv
    from matplotlib import pyplot as plt
    from ipywidgets import FileUpload

    # Fungsi untuk menampilkan gambar setelah diupload
    def process_image(change):
        # Mengambil file yang diupload
        uploaded_image = list(uploaded_files.value.values())[0]

        # Mengkonversi gambar ke format numpy array
        img_array = np.frombuffer(uploaded_image['content'], np.uint8)
        img = cv.imdecode(img_array, cv.IMREAD_COLOR)

        # Memastikan gambar berhasil dibaca
        assert img is not None, "File could not be read."

        # Membuat kernel filter rata-rata dengan ukuran yang lebih besar (misalnya 9x9)
        kernel = np.ones((9, 9), np.float32) / 81  # 9x9, dan jumlahnya adalah 1/81
        dst = cv.filter2D(img, -1, kernel)

        # Menampilkan gambar asli dan hasil filter
        plt.subplot(121), plt.imshow(cv.cvtColor(img, cv.COLOR_BGR2RGB)), plt.title('Original')
        plt.xticks([]), plt.yticks([])
        plt.subplot(122), plt.imshow(cv.cvtColor(dst, cv.COLOR_BGR2RGB)), plt.title('Averaging')
        plt.xticks([]), plt.yticks([])
        plt.show()

    # Membuat widget upload file
    uploaded_files = FileUpload(accept='image/*', multiple=False)
    uploaded_files.observe(process_image, names='value')
    display(uploaded_files)
```

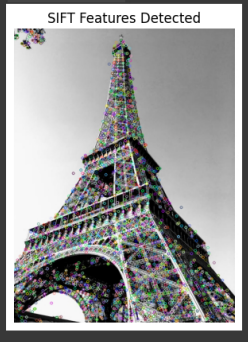
Upload (1)



Deteksi Fitur SIFT

```
[10] import cv2
     import numpy as np
     from matplotlib import pyplot as plt
     from ipywidgets import FileUpload
     from io import BytesIO

     # Fungsi untuk menampilkan gambar setelah diupload
     def process_image(change):
         # Mengambil file yang diupload
         uploaded_image = list(uploaded_files.value.values())[0]

         # Membaca gambar sebagai array numpy
         img_array = np.frombuffer(uploaded_image['content'], np.uint8)
         img = cv2.imdecode(img_array, cv2.IMREAD_COLOR)

         # Mengonversi gambar ke grayscale
         gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

         # Membuat SIFT feature extractor
         sift = cv2.SIFT_create()

         # Mendeteksi fitur pada gambar
         keypoints, descriptors = sift.detectAndCompute(gray, None)

         # Menggambar keypoints yang terdeteksi pada gambar
         sift_image = cv2.drawKeypoints(img, keypoints, None)

         # Menampilkan gambar hasil deteksi
         plt.imshow(cv2.cvtColor(sift_image, cv2.COLOR_BGR2RGB))
         plt.title('SIFT Features Detected')
         plt.axis('off')
         plt.show()

     # Membuat widget upload file
     uploaded_files = FileUpload(accept='image/*', multiple=False)
     uploaded_files.observe(process_image, names='value')
     display(uploaded_files)
```

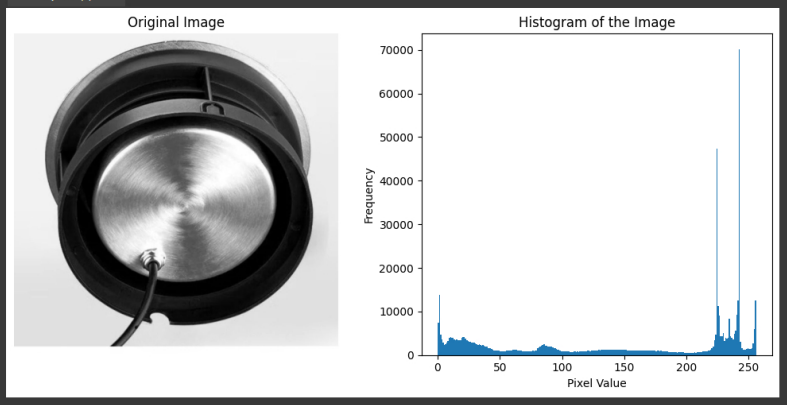
Upload (1)



Representasi Histogram

```python
import cv2
from matplotlib import pyplot as plt
import ipywidgets as widgets
from IPython.display import display

# Membuat file picker
uploader = widgets.FileUpload(accept='image/*', multiple=False)
display(uploader)

def process_file(change):
    if uploader.value:
        file_data = list(uploader.value.values())[0]
        content = file_data['content']

        # Membaca gambar dari file yang diunggah
        with open("uploaded_image.jpg", "wb") as f:
            f.write(content)

        img = cv2.imread("uploaded_image.jpg", 0)
        if img is not None:
            # Membuat subplot untuk menampilkan gambar dan histogram
            plt.figure(figsize=(10, 5))

            # Menampilkan gambar original
            plt.subplot(1, 2, 1)
            plt.imshow(img, cmap='gray')
            plt.title('Original Image')
            plt.axis('off')

            # Menampilkan histogram
            plt.subplot(1, 2, 2)
            plt.hist(img.ravel(), 256, [0, 256])
            plt.title('Histogram of the Image')
            plt.xlabel('Pixel Value')
            plt.ylabel('Frequency')

            # Menampilkan kedua plot
            plt.tight_layout()
            plt.show()
        else:
            print("Gagal membaca file gambar.")

# Memproses file setelah diunggah
uploader.observe(process_file, names='value')
```

Upload (1)

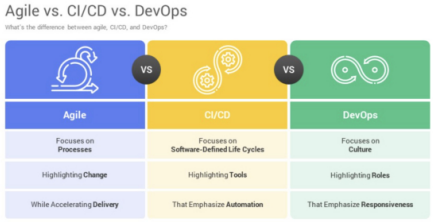


Gaussian Smoothing

```python
from PIL import Image, ImageFilter
import matplotlib.pyplot as plt
import ipywidgets as widgets
from IPython.display import display

# Membuat file picker
uploader = widgets.FileUpload(accept='image/*', multiple=False)
display(uploader)

def process_file(change):
    if uploader.value:
        file_data = list(uploader.value.values())[0]
        content = file_data['content']

        # Membaca gambar dari file yang diunggah
        with open("uploaded_image.jpg", "wb") as f:
            f.write(content)

        # Membuka gambar asli
        original_image = Image.open("uploaded_image.jpg")

        # Mengaplikasikan filter Gaussian Blur dengan radius tetap
        radius = 4
        blurred_image = original_image.filter(ImageFilter.GaussianBlur(radius=radius))

        # Menampilkan gambar asli dan hasil filter secara berdampingan
        plt.figure(figsize=(12, 6))

        # Menampilkan gambar asli
        plt.subplot(1, 2, 1)
        plt.imshow(original_image)
        plt.title("Original Image")
        plt.axis('off')

        # Menampilkan gambar hasil filter
        plt.subplot(1, 2, 2)
        plt.imshow(blurred_image)
        plt.title(f"Gaussian Blurred Image (Radius: {radius})")
        plt.axis('off')

        # Menampilkan kedua gambar
        plt.tight_layout()
        plt.show()
    else:
        print("Tidak ada file yang diunggah.")

# Memproses file setelah diunggah
uploader.observe(process_file, names='value')
```

Upload (1)

Original Image

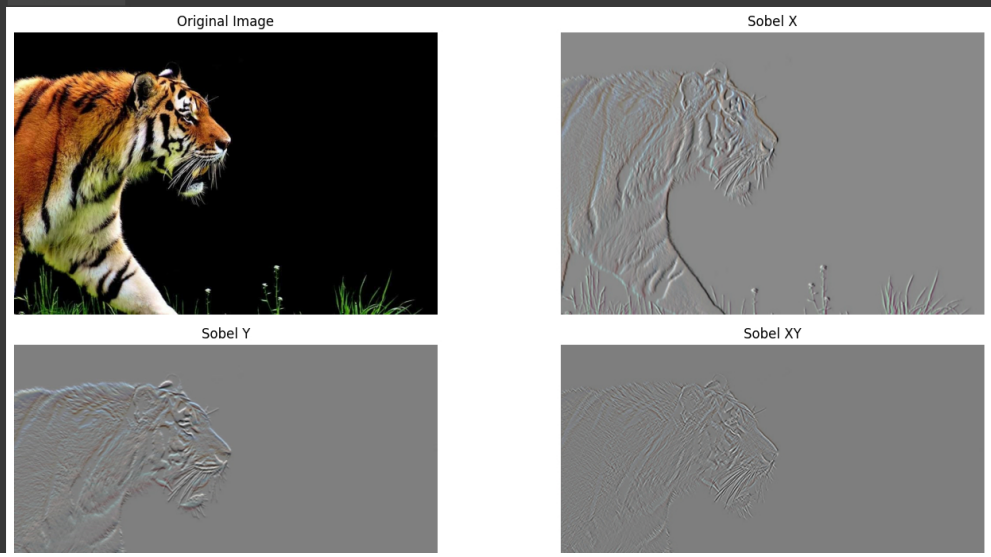


Gaussian Blurred Image (Radius: 4)

Edge detection with Sobel

```
[19] import cv2
     import matplotlib.pyplot as plt
     import ipywidgets as widgets
     from IPython.display import display

     # Membuat file picker
     uploader = widgets.FileUpload(accept='image/*', multiple=False)
     display(uploader)

     def process_file(change):
         if uploader.value:
             file_data = list(uploader.value.values())[0]
             content = file_data['content']

             # Membaca gambar dari file yang diunggah
             with open("uploaded_image.jpg", "wb") as f:
                 f.write(content)

             # Membaca gambar menggunakan OpenCV
             img = cv2.imread("uploaded_image.jpg")
             img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)  # Konversi ke RGB untuk matplotlib

             # Blur untuk deteksi tepi yang lebih baik
             img_blur = cv2.GaussianBlur(img, (3, 3), 0)

             # Sobel Edge Detection
             sobelx = cv2.Sobel(src=img_blur, ddepth=cv2.CV_64F, dx=1, dy=0, ksize=5)  # Sobel X
             sobely = cv2.Sobel(src=img_blur, ddepth=cv2.CV_64F, dx=0, dy=1, ksize=5)  # Sobel Y
             sobelxy = cv2.Sobel(src=img_blur, ddepth=cv2.CV_64F, dx=1, dy=1, ksize=5)  # Sobel XY

             # Normalisasi hasil Sobel ke rentang [0, 255]
             sobelx_norm = cv2.normalize(sobelx, None, 0, 255, cv2.NORM_MINMAX).astype('uint8')
             sobely_norm = cv2.normalize(sobely, None, 0, 255, cv2.NORM_MINMAX).astype('uint8')
             sobelxy_norm = cv2.normalize(sobelxy, None, 0, 255, cv2.NORM_MINMAX).astype('uint8')

             # Menampilkan hasil
             plt.figure(figsize=(15, 8))

             # Gambar asli
             plt.subplot(2, 2, 1)
             plt.imshow(img_rgb)
             plt.title("Original Image")
             plt.axis('off')

             # Sobel X
             plt.subplot(2, 2, 2)
             plt.imshow(sobelx_norm, cmap='gray')
             plt.title("Sobel X")
             plt.axis('off')

             # Sobel Y
             plt.subplot(2, 2, 3)
             plt.imshow(sobely_norm, cmap='gray')
             plt.title("Sobel Y")
             plt.axis('off')

             # Sobel XY
             plt.subplot(2, 2, 4)
             plt.imshow(sobelxy_norm, cmap='gray')
             plt.title("Sobel XY")
             plt.axis('off')

             # Tampilkan semua gambar
             plt.tight_layout()
             plt.show()
         else:
             print("Tidak ada file yang diunggah.")

     # Memproses file setelah diunggah
     uploader.observe(process_file, names='value')
```

Upload (1)



Histogram of Oriented Gradients (HOG)

```python
import matplotlib.pyplot as plt
from skimage.feature import hog
from skimage import exposure, io
import ipywidgets as widgets
from IPython.display import display

# Membuat widget untuk upload file
uploader = widgets.FileUpload(accept='image/*', multiple=False)
display(uploader)

def process_file(change):
    if uploader.value:
        # Membaca file dari uploader
        file_data = list(uploader.value.values())[0]
        content = file_data['content']

        # Membaca gambar menggunakan skimage
        with open("uploaded_image.jpg", "wb") as f:
            f.write(content)
        image = io.imread("uploaded_image.jpg")

        # Menghitung HOG
        fd, hog_image = hog(
            image,
            orientations=8,
            pixels_per_cell=(16, 16),
            cells_per_block=(1, 1),
            visualize=True,
            channel_axis=-1,
        )

        # Rescale histogram for better display
        hog_image_rescaled = exposure.rescale_intensity(hog_image, in_range=(0, 10))

        # Menampilkan gambar asli dan HOG
        fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 6), sharex=True, sharey=True)

        ax1.axis('off')
        ax1.imshow(image, cmap=plt.cm.gray)
        ax1.set_title('Input Image')

        ax2.axis('off')
        ax2.imshow(hog_image_rescaled, cmap=plt.cm.gray)
        ax2.set_title('Histogram of Oriented Gradients')

        plt.show()
    else:
        print("Tidak ada file yang diunggah.")

# Menghubungkan uploader dengan fungsi proses
uploader.observe(process_file, names='value')
```

Upload (1)