Name: Vivek Vitthal Avhad (3059)

In [1]:

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
```

In [2]:

```
image = cv2.imread('Ex1.jpg')
```

In [3]:

```
image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
```

In [4]:

```
avg_blur = cv2.blur(image, (5, 5))
```

In [5]:

```
gaussian_blur = cv2.GaussianBlur(image, (5, 5), 0)
```

In [6]:

```
median_blur = cv2.medianBlur(image, 5)
```

In [7]:

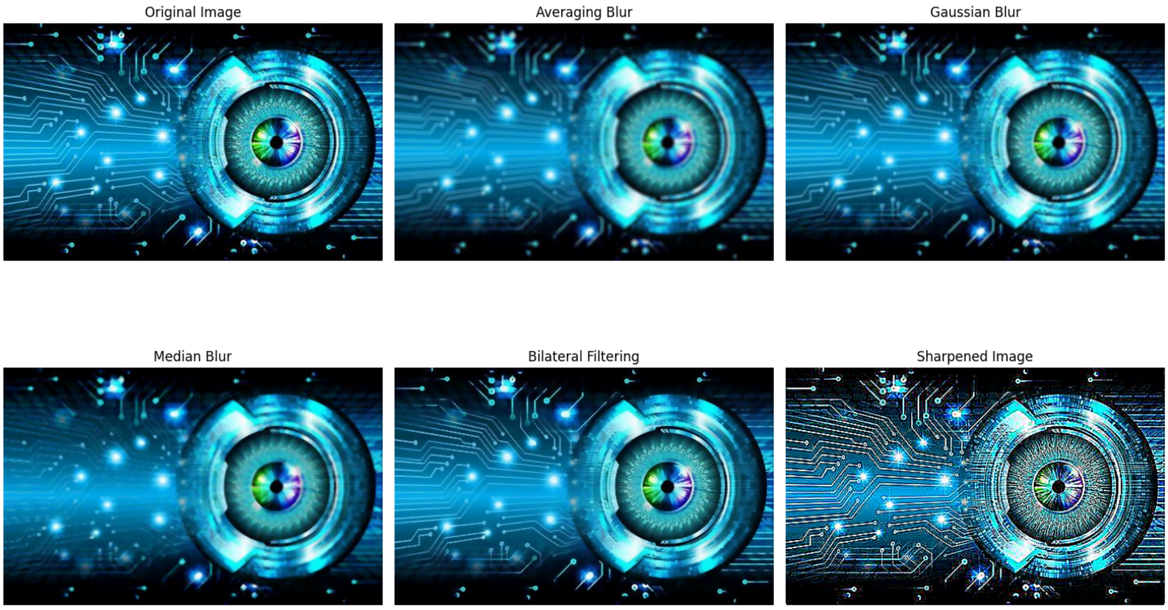
```
bilateral_blur = cv2.bilateralFilter(image, 9, 75, 75)
```

In [8]:

```
sharpening_kernel = np.array([[-1, -1, -1],
[-1, 9, -1],
[-1, -1, -1]])

sharpened = cv2.filter2D(image, -1, sharpening_kernel)
```

In [9]:

```
fig, ax = plt.subplots(2, 3, figsize=(15, 10))

ax[0, 0].imshow(image)
ax[0, 0].set_title("Original Image")
ax[0, 0].axis("off")

ax[0, 1].imshow(avg_blur)
ax[0, 1].set_title("Averaging Blur")
ax[0, 1].axis("off")

ax[0, 2].imshow(gaussian_blur)
ax[0, 2].set_title("Gaussian Blur")
ax[0, 2].axis("off")

ax[1, 0].imshow(median_blur)
ax[1, 0].set_title("Median Blur")
ax[1, 0].axis("off")

ax[1, 1].imshow(bilateral_blur)
ax[1, 1].set_title("Bilateral Filtering")
ax[1, 1].axis("off")

ax[1, 2].imshow(sharpened)
ax[1, 2].set_title("Sharpened Image")
ax[1, 2].axis("off")
plt.tight_layout()
plt.show()
```



1. Apply a bilateral filter to preserve edges while smoothing an image.

In [10]:
```
image2 = cv2.imread('Que.png')
```

In [11]:
```
bilateral_filtered_image = cv2.bilateralFilter(image2, d=9, sigmaColor=75, sigma
```

In [12]:
```
plt.imshow(cv2.cvtColor(bilateral_filtered_image, cv2.COLOR_BGR2RGB))
plt.axis('off')  # Hide the axis
#plt.show()
```

Out[12]:
```
(-0.5, 929.5, 520.5, -0.5)
```

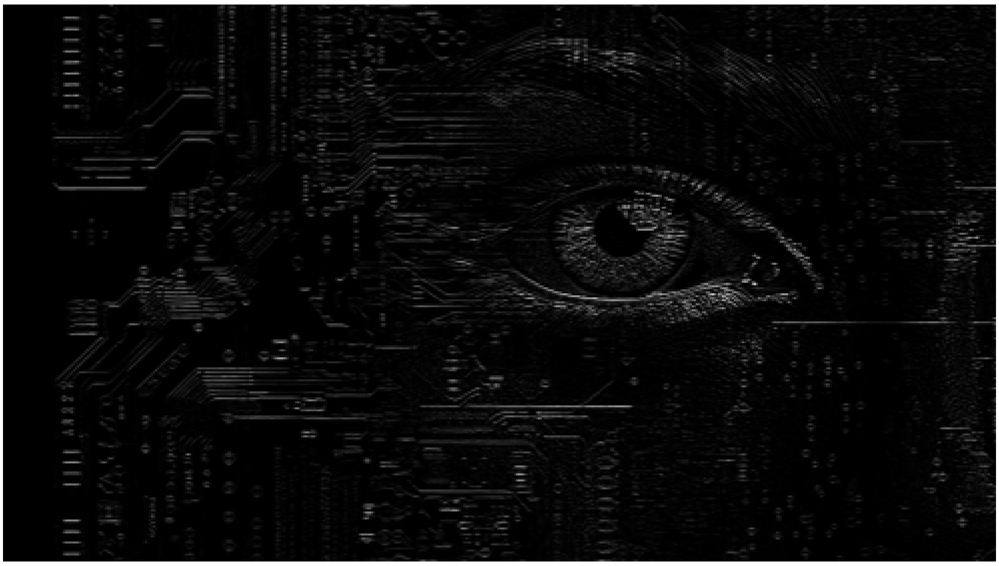
2. Create a custom kernel to detect horizontal edges in an image.

In [13]:
```
image3 = cv2.imread('Que.png', cv2.IMREAD_GRAYSCALE)
```

In [14]:
```
kernel = np.array([[-1, -1, -1],
                   [ 2,  2,  2],
                   [-1, -1, -1]])
horizontal_edges = cv2.filter2D(image3, -1, kernel)
```

In [15]:
```
plt.imshow(cv2.cvtColor(horizontal_edges, cv2.COLOR_BGR2RGB))
plt.axis('off')
```

Out[15]:
```
(-0.5, 929.5, 520.5, -0.5)
```

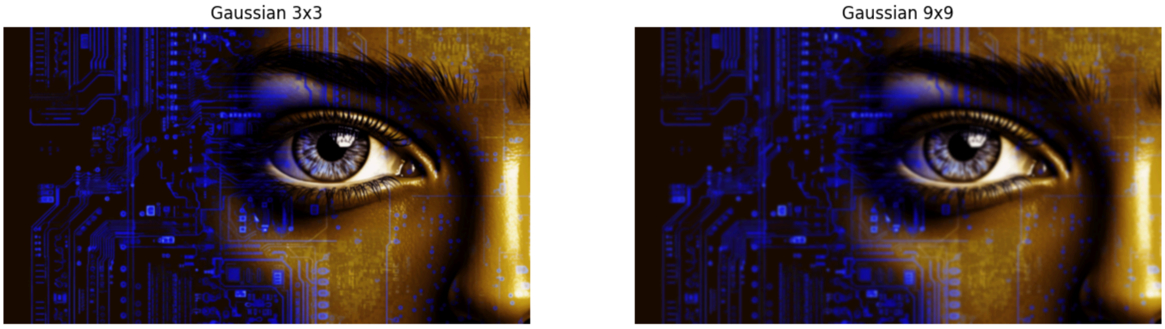


3. Compare the effects of different kernel sizes in Gaussian filtering.

In [16]:
```
gaussian_3x3 = cv2.GaussianBlur(image2, (3, 3), 0)
gaussian_9x9 = cv2.GaussianBlur(image2, (9, 9), 0)
```

In [17]:
```
fig, ax = plt.subplots(1, 2, figsize=(15, 10))

ax[0].imshow(gaussian_3x3)
ax[0].set_title("Gaussian 3x3")
ax[0].axis("off")

ax[1].imshow(gaussian_9x9)
ax[1].set_title("Gaussian 9x9")
ax[1].axis("off")
```

Out[17]:
```
(-0.5, 929.5, 520.5, -0.5)
```



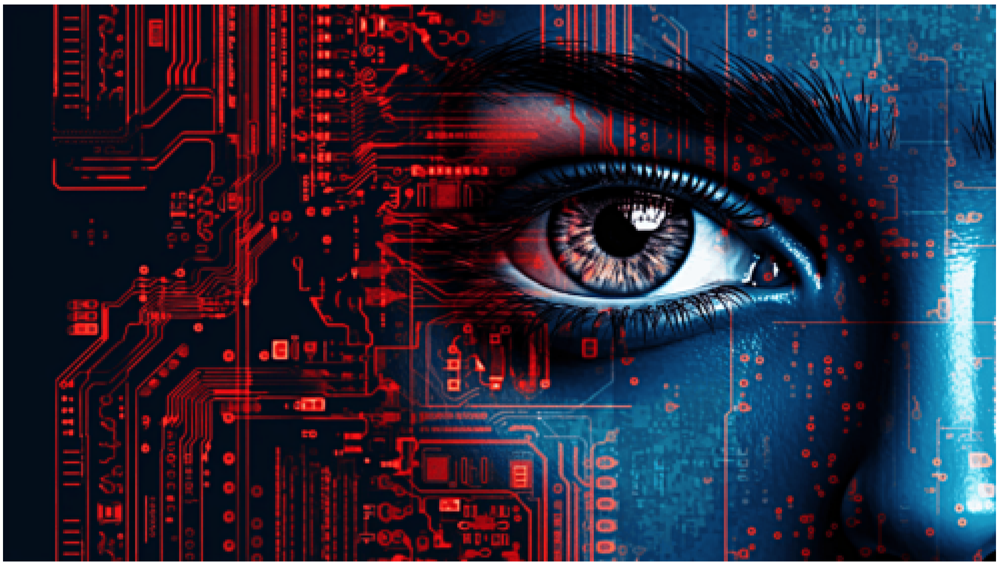
4. Implement image sharpening using the Unsharp Masking technique.

In [18]:
```
gaussian_blur = cv2.GaussianBlur(image2, (9, 9), 10.0)
```

```
unsharp_image = cv2.addWeighted(image2, 1.5, gaussian_blur, -0.5, 0)
```

In [19]:
```
plt.imshow(cv2.cvtColor(unsharp_image, cv2.COLOR_BGR2RGB))
plt.axis('off')
```

Out[19]:
```
(-0.5, 929.5, 520.5, -0.5)
```



5. Perform adaptive thresholding after applying smoothing filters.

In [20]:
```
image4 = cv2.imread('Que.png', cv2.IMREAD_GRAYSCALE)
```

In [21]:
```
smoothed_image = cv2.GaussianBlur(image4, (5, 5), 0)
adaptive_thresh_image = cv2.adaptiveThreshold(smoothed_image, 255, cv2.ADAPTIVE_
```

In [22]:
```
plt.imshow(cv2.cvtColor(adaptive_thresh_image, cv2.COLOR_BGR2RGB))
plt.axis('off')
```

Out[22]:
```
(-0.5, 929.5, 520.5, -0.5)
```