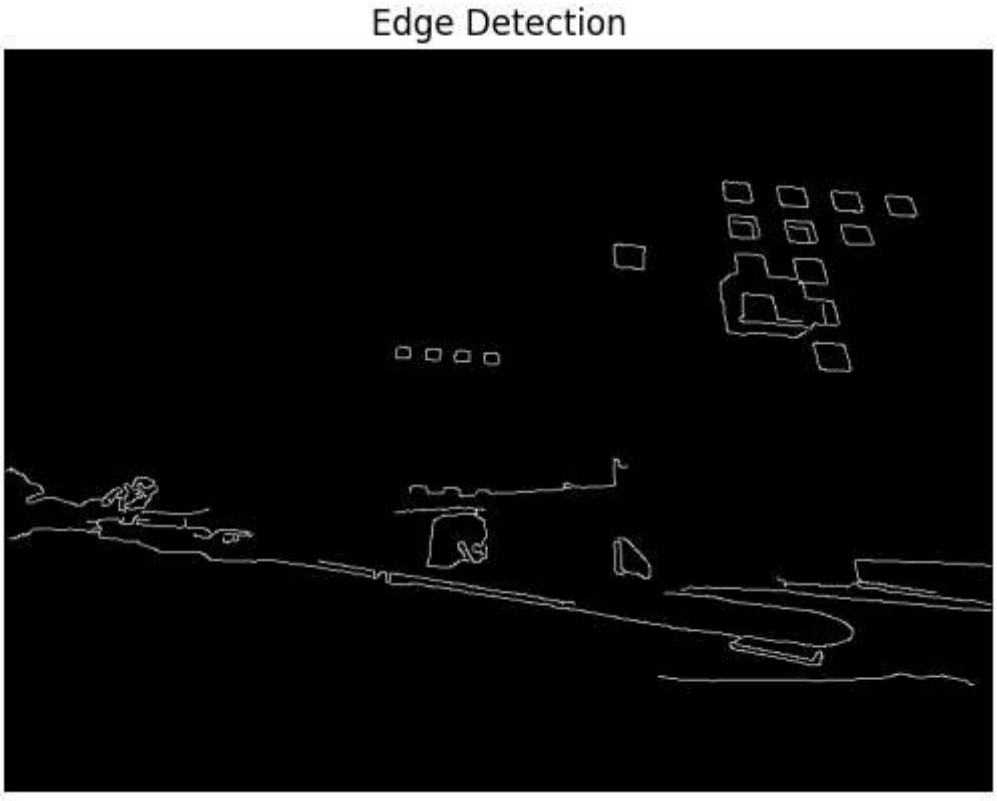
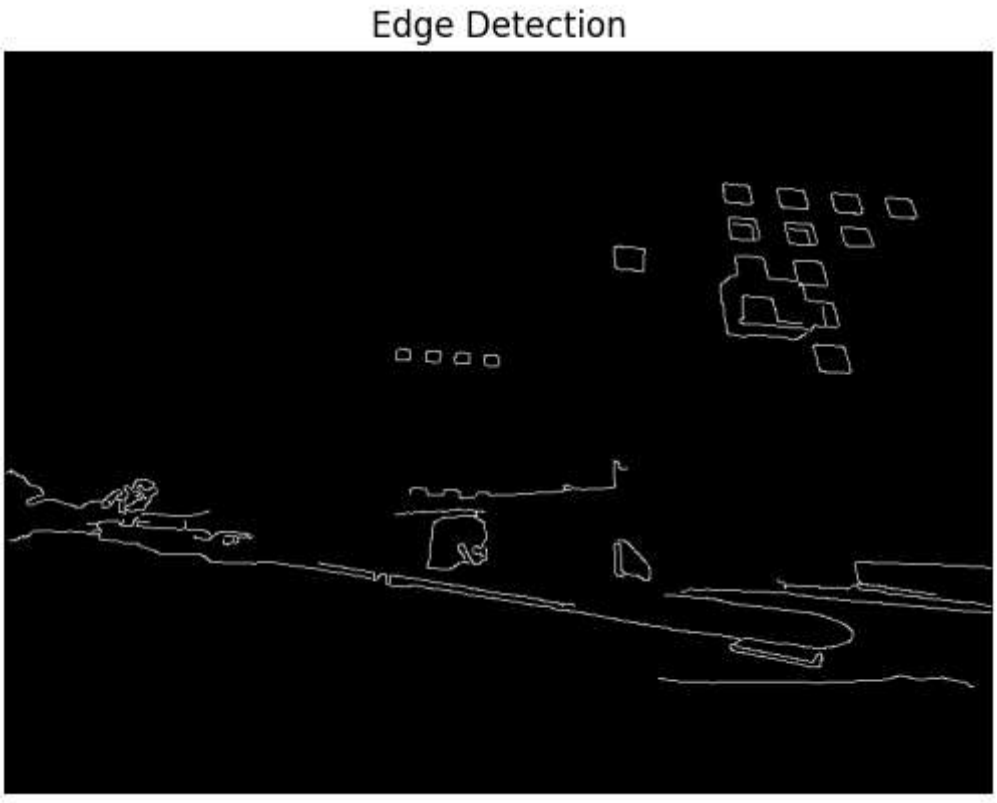
at sigma=1.0, low_threshold=0.1, high_threshold=0.9

In [30]:
```
import numpy as np
import matplotlib.pyplot as plt
from skimage import io, color, filters, feature
from scipy import ndimage


#for loading the image
image = io.imread('ct1.jpg')

downscale_factor = 2

# Perform datapreprocessing degrading resolution of picture
downscaled_image = ndimage.zoom(image, (1/downscale_factor, 1/downscale_factor, 1),

#converting image to grayscale
image_gray = color.rgb2gray(downscaled_image)

#apply gaussian blur to image (reducing noise)
blur_image = filters.gaussian(image_gray, sigma=1.0)

#calculate the gradient of the image using Sobel operator

sobel_x = filters.sobel(blur_image, axis=1)  #Gradient is the rate of change of ima
sobel_y = filters.sobel(blur_image, axis=0)

edges = feature.canny(blur_image, sigma=1.0, low_threshold=0.1, high_threshold=0.9)

#Display original iamge
plt.imshow(image)
plt.title('Original Image')
plt.axis('off')
plt.show()

#display image with edges
plt.imshow(edges, cmap='gray')
plt.title('Edge Detection')
plt.axis('off')
plt.show()
```

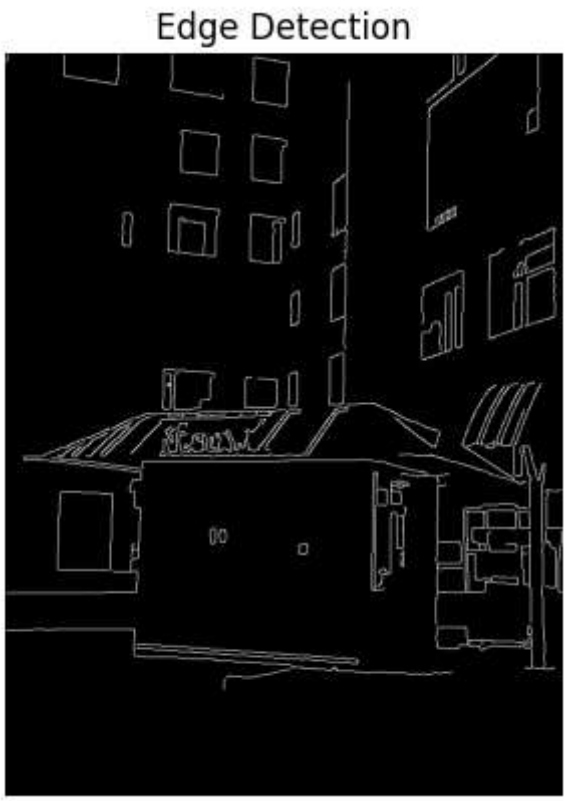


In [29]:



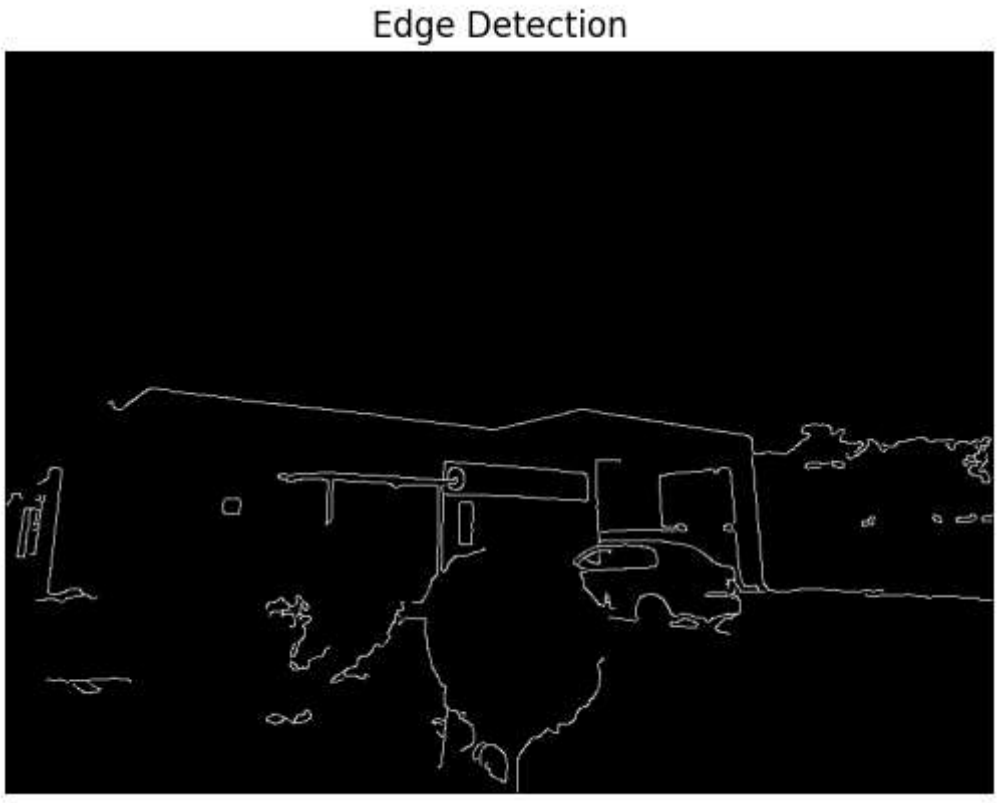
In [31]:
```
import numpy as np
import matplotlib.pyplot as plt
from skimage import io, color, filters, feature
from scipy import ndimage
```

```
#for loading the image
image = io.imread('ct2.jpg')

downscale_factor = 2

# Perform datapreprocessing degrading resolution of picture
downscaled_image = ndimage.zoom(image, (1/downscale_factor, 1/downscale_factor, 1),

#converting image to grayscale
image_gray = color.rgb2gray(downscaled_image)

#apply gaussian blur to image (reducing noise)
blur_image = filters.gaussian(image_gray, sigma=1.0)

#calculate the gradient of the image using Sobel operator

sobel_x = filters.sobel(blur_image, axis=1)  #Gradient is the rate of change of ima
sobel_y = filters.sobel(blur_image, axis=0)

edges = feature.canny(blur_image, sigma=1.0, low_threshold=0.1, high_threshold=0.9)

#Display original iamge
plt.imshow(image)
plt.title('Original Image')
plt.axis('off')
plt.show()

#display image with edges
plt.imshow(edges, cmap='gray')
plt.title('Edge Detection')
plt.axis('off')
plt.show()
```



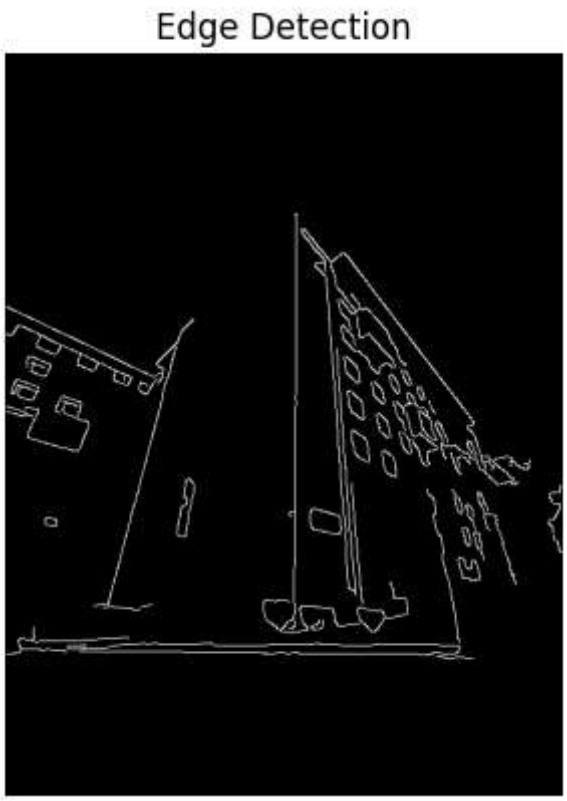
In [32]:
```
import numpy as np
import matplotlib.pyplot as plt
from skimage import io, color, filters, feature
from scipy import ndimage
```

```
#for loading the image
image = io.imread('ct3.jpg')

downscale_factor = 2

# Perform datapreprocessing degrading resolution of picture
downscaled_image = ndimage.zoom(image, (1/downscale_factor, 1/downscale_factor, 1),

#converting image to grayscale
image_gray = color.rgb2gray(downscaled_image)

#apply gaussian blur to image (reducing noise)
blur_image = filters.gaussian(image_gray, sigma=1.0)

#calculate the gradient of the image using Sobel operator

sobel_x = filters.sobel(blur_image, axis=1)  #Gradient is the rate of change of ima
sobel_y = filters.sobel(blur_image, axis=0)

edges = feature.canny(blur_image, sigma=1.0, low_threshold=0.1, high_threshold=0.9)

#Display original iamge
plt.imshow(image)
plt.title('Original Image')
plt.axis('off')
plt.show()

#display image with edges
plt.imshow(edges, cmap='gray')
plt.title('Edge Detection')
plt.axis('off')
plt.show()
```



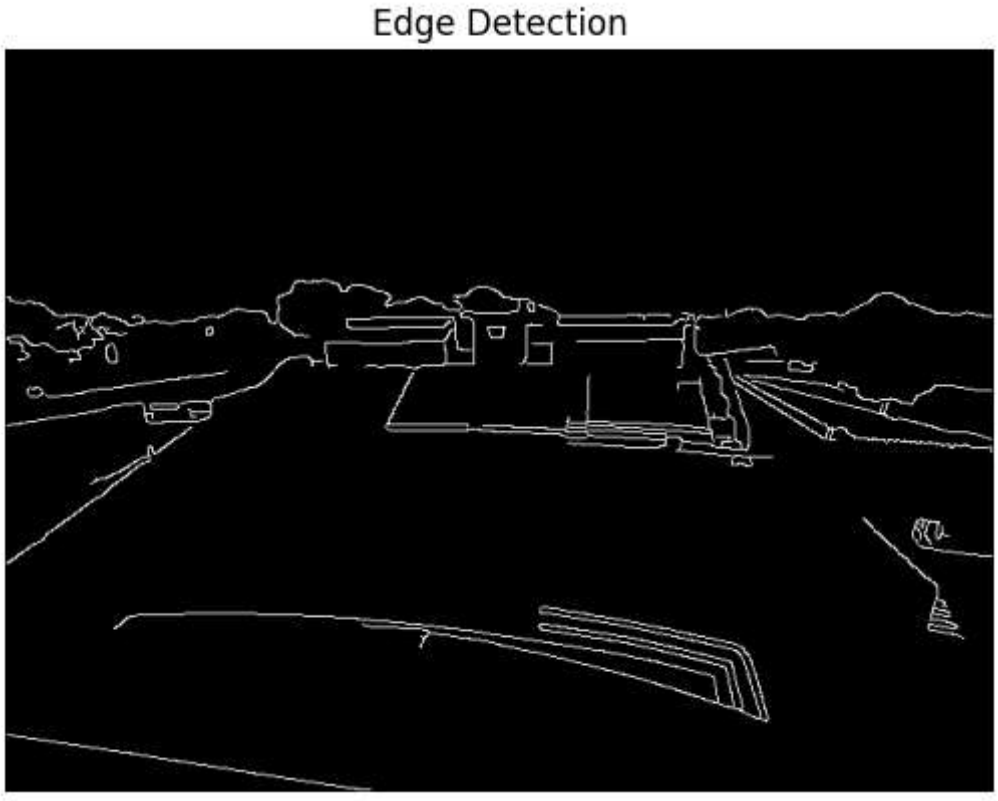
In [33]:
```
import numpy as np
import matplotlib.pyplot as plt
from skimage import io, color, filters, feature
from scipy import ndimage
```

```
#for loading the image
image = io.imread('ct4.jpg')

downscale_factor = 2

# Perform datapreprocessing degrading resolution of picture
downscaled_image = ndimage.zoom(image, (1/downscale_factor, 1/downscale_factor, 1),

#converting image to grayscale
image_gray = color.rgb2gray(downscaled_image)

#apply gaussian blur to image (reducing noise)
blur_image = filters.gaussian(image_gray, sigma=1.0)

#calculate the gradient of the image using Sobel operator

sobel_x = filters.sobel(blur_image, axis=1)  #Gradient is the rate of change of ima
sobel_y = filters.sobel(blur_image, axis=0)

edges = feature.canny(blur_image, sigma=1.0, low_threshold=0.1, high_threshold=0.9)

#Display original iamge
plt.imshow(image)
plt.title('Original Image')
plt.axis('off')
plt.show()

#display image with edges
plt.imshow(edges, cmap='gray')
plt.title('Edge Detection')
plt.axis('off')
plt.show()
```



In [34]:
```
import numpy as np
import matplotlib.pyplot as plt
from skimage import io, color, filters, feature
from scipy import ndimage
```

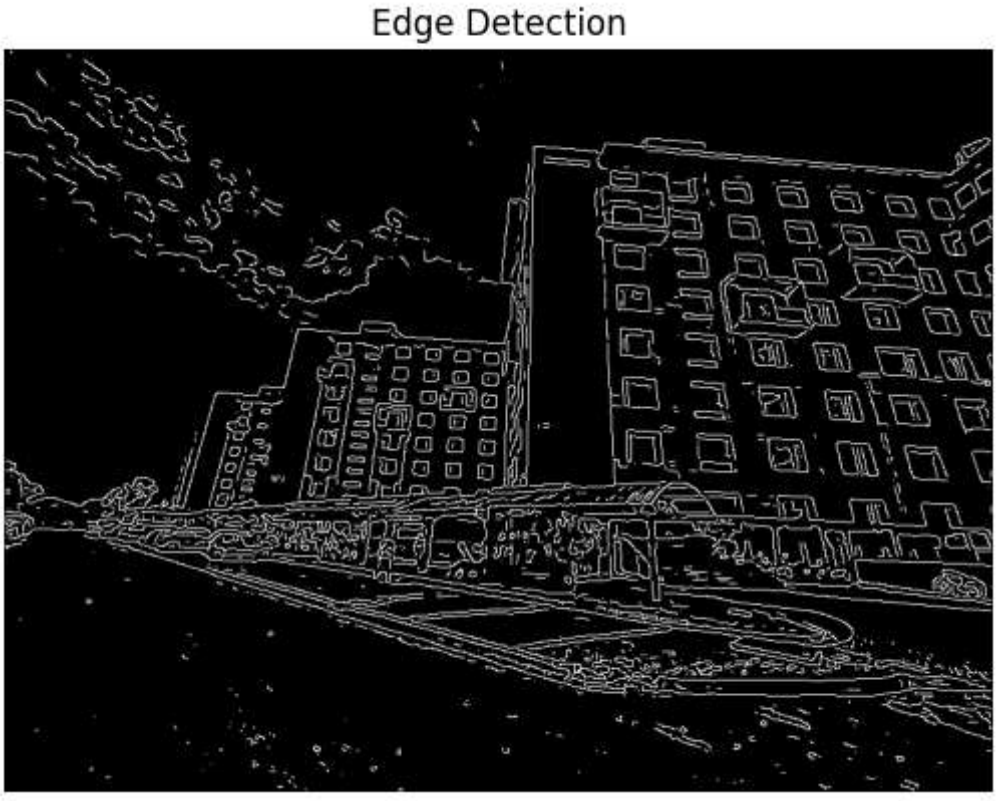
```
#for loading the image
image = io.imread('ct5.jpg')

downscale_factor = 2

# Perform datapreprocessing degrading resolution of picture
downscaled_image = ndimage.zoom(image, (1/downscale_factor, 1/downscale_factor, 1),

#converting image to grayscale
image_gray = color.rgb2gray(downscaled_image)

#apply gaussian blur to image (reducing noise)
blur_image = filters.gaussian(image_gray, sigma=1.0)

#calculate the gradient of the image using Sobel operator

sobel_x = filters.sobel(blur_image, axis=1)  #Gradient is the rate of change of ima
sobel_y = filters.sobel(blur_image, axis=0)

edges = feature.canny(blur_image, sigma=1.0, low_threshold=0.1, high_threshold=0.9)

#Display original iamge
plt.imshow(image)
plt.title('Original Image')
plt.axis('off')
plt.show()

#display image with edges
plt.imshow(edges, cmap='gray')
plt.title('Edge Detection')
plt.axis('off')
plt.show()
```



In [ ]:
```
# decreasing threshold value to 0.1 ----> sigma=1.0, low_threshold=0.1, high_thresh
```

In [35]:
```
import numpy as np
import matplotlib.pyplot as plt
from skimage import io, color, filters, feature
from scipy import ndimage
```

```
#for loading the image
image = io.imread('ct1.jpg')

downscale_factor = 2

# Perform datapreprocessing degrading resolution of picture
downscaled_image = ndimage.zoom(image, (1/downscale_factor, 1/downscale_factor, 1),

#converting image to grayscale
image_gray = color.rgb2gray(downscaled_image)

#apply gaussian blur to image (reducing noise)
blur_image = filters.gaussian(image_gray, sigma=1.0)

#calculate the gradient of the image using Sobel operator

sobel_x = filters.sobel(blur_image, axis=1)  #Gradient is the rate of change of ima
sobel_y = filters.sobel(blur_image, axis=0)

edges = feature.canny(blur_image, sigma=1.0, low_threshold=0.1, high_threshold=0.1)

#Display original iamge
plt.imshow(image)
plt.title('Original Image')
plt.axis('off')
plt.show()

#display image with edges
plt.imshow(edges, cmap='gray')
plt.title('Edge Detection')
plt.axis('off')
plt.show()
```

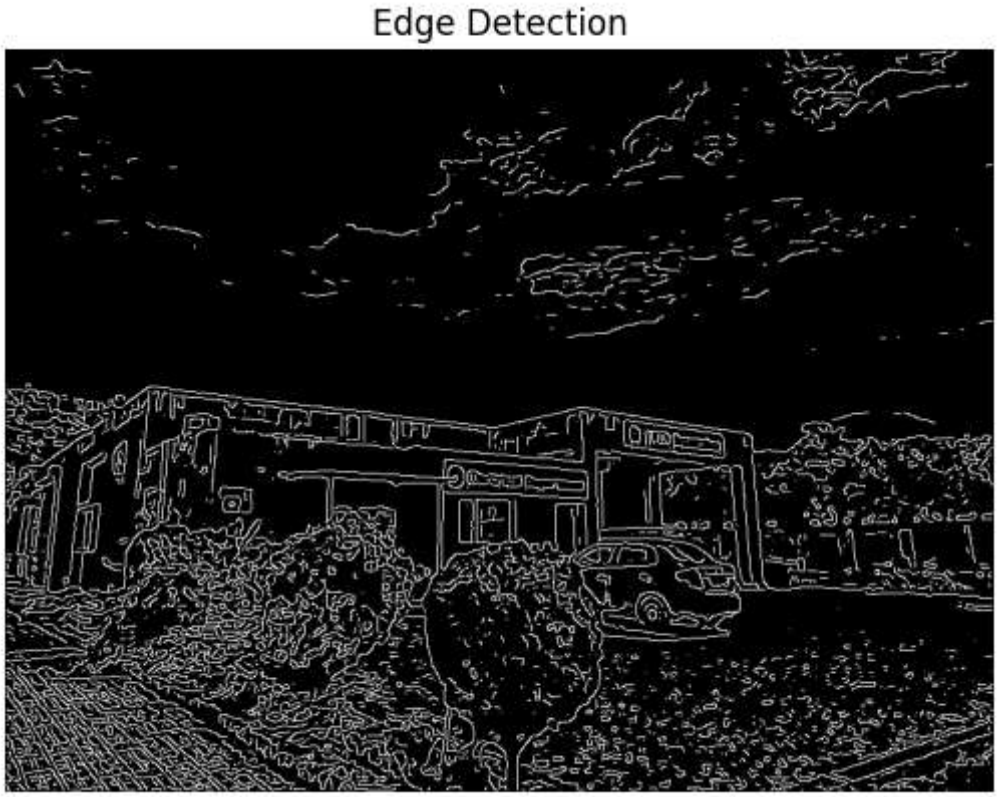


In [36]:
```
import numpy as np
import matplotlib.pyplot as plt
from skimage import io, color, filters, feature
from scipy import ndimage
```

```
#for loading the image
image = io.imread('ct2.jpg')

downscale_factor = 2

# Perform datapreprocessing degrading resolution of picture
downscaled_image = ndimage.zoom(image, (1/downscale_factor, 1/downscale_factor, 1),

#converting image to grayscale
image_gray = color.rgb2gray(downscaled_image)

#apply gaussian blur to image (reducing noise)
blur_image = filters.gaussian(image_gray, sigma=1.0)

#calculate the gradient of the image using Sobel operator

sobel_x = filters.sobel(blur_image, axis=1)  #Gradient is the rate of change of ima
sobel_y = filters.sobel(blur_image, axis=0)

edges = feature.canny(blur_image, sigma=1.0, low_threshold=0.1, high_threshold=0.1)

#Display original iamge
plt.imshow(image)
plt.title('Original Image')
plt.axis('off')
plt.show()

#display image with edges
plt.imshow(edges, cmap='gray')
plt.title('Edge Detection')
plt.axis('off')
plt.show()
```

In [37]:
```
import numpy as np
import matplotlib.pyplot as plt
from skimage import io, color, filters, feature
from scipy import ndimage
```

```
#for loading the image
image = io.imread('ct3.jpg')

downscale_factor = 2

# Perform datapreprocessing degrading resolution of picture
downscaled_image = ndimage.zoom(image, (1/downscale_factor, 1/downscale_factor, 1),

#converting image to grayscale
image_gray = color.rgb2gray(downscaled_image)

#apply gaussian blur to image (reducing noise)
blur_image = filters.gaussian(image_gray, sigma=1.0)

#calculate the gradient of the image using Sobel operator

sobel_x = filters.sobel(blur_image, axis=1)  #Gradient is the rate of change of ima
sobel_y = filters.sobel(blur_image, axis=0)

edges = feature.canny(blur_image, sigma=1.0, low_threshold=0.1, high_threshold=0.1)

#Display original iamge
plt.imshow(image)
plt.title('Original Image')
plt.axis('off')
plt.show()

#display image with edges
plt.imshow(edges, cmap='gray')
plt.title('Edge Detection')
plt.axis('off')
plt.show()
```

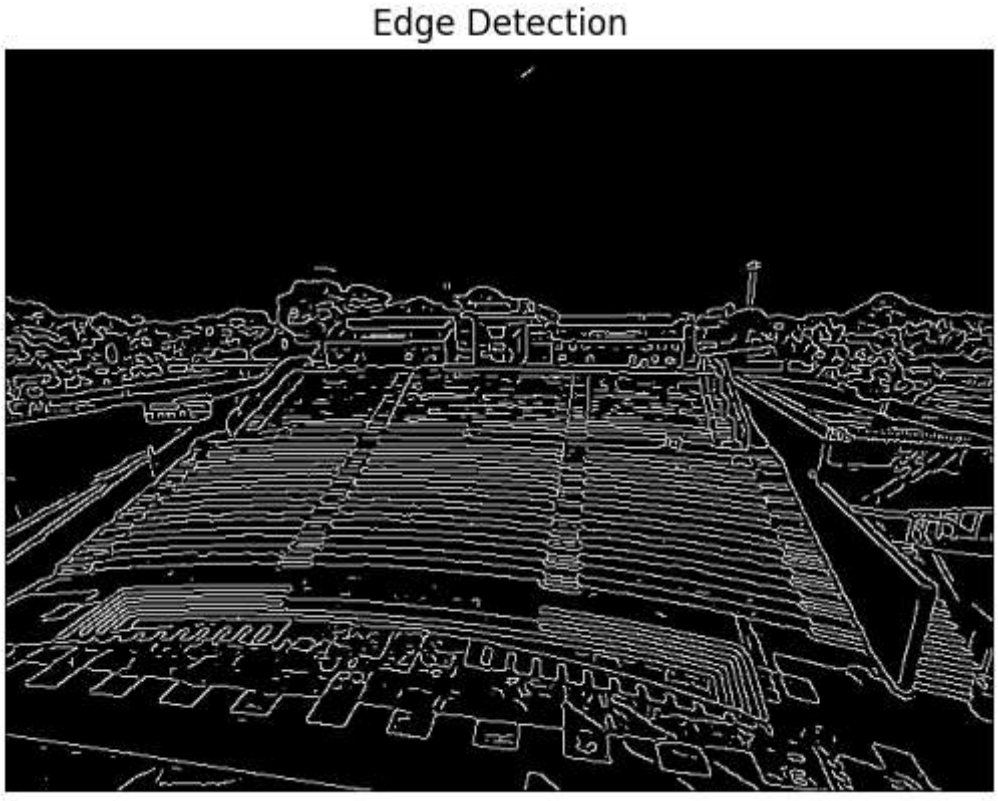


In [38]:
```
import numpy as np
import matplotlib.pyplot as plt
from skimage import io, color, filters, feature
from scipy import ndimage
```

```
#for loading the image
image = io.imread('ct4.jpg')

downscale_factor = 2

# Perform datapreprocessing degrading resolution of picture
downscaled_image = ndimage.zoom(image, (1/downscale_factor, 1/downscale_factor, 1),

#converting image to grayscale
image_gray = color.rgb2gray(downscaled_image)

#apply gaussian blur to image (reducing noise)
blur_image = filters.gaussian(image_gray, sigma=1.0)

#calculate the gradient of the image using Sobel operator

sobel_x = filters.sobel(blur_image, axis=1)  #Gradient is the rate of change of ima
sobel_y = filters.sobel(blur_image, axis=0)

edges = feature.canny(blur_image, sigma=1.0, low_threshold=0.1, high_threshold=0.1)

#Display original iamge
plt.imshow(image)
plt.title('Original Image')
plt.axis('off')
plt.show()

#display image with edges
plt.imshow(edges, cmap='gray')
plt.title('Edge Detection')
plt.axis('off')
plt.show()
```

In [39]:
```
import numpy as np
import matplotlib.pyplot as plt
from skimage import io, color, filters, feature
from scipy import ndimage
```

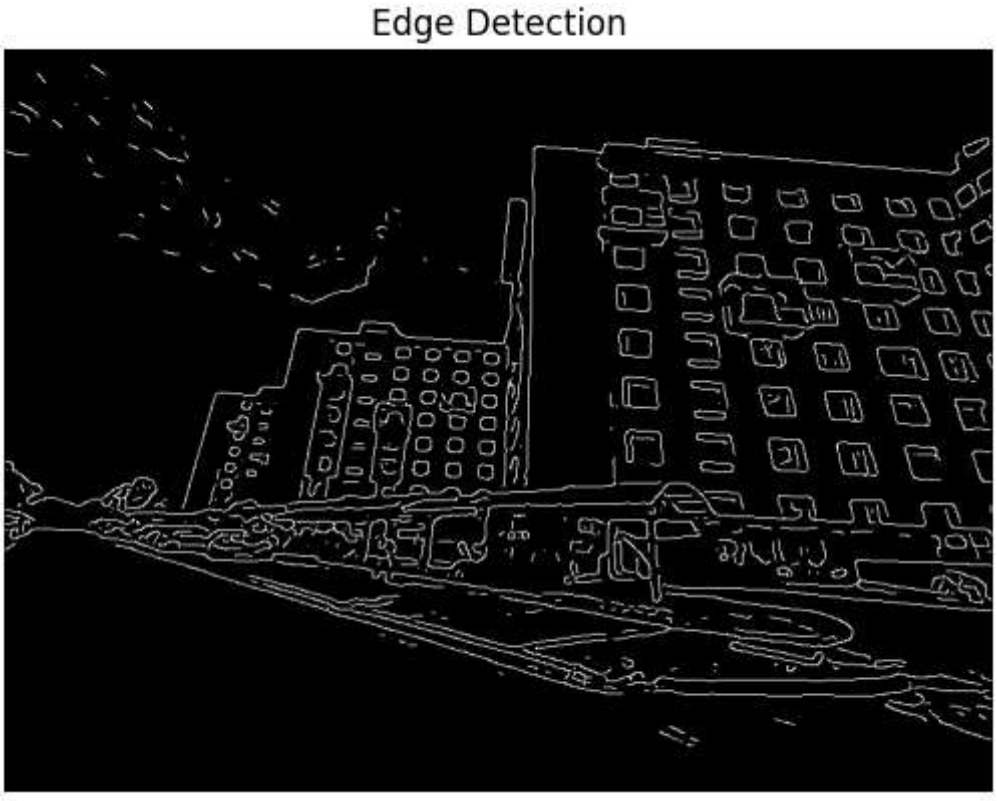
```
#for loading the image
image = io.imread('ct5.jpg')

downscale_factor = 2

# Perform datapreprocessing degrading resolution of picture
downscaled_image = ndimage.zoom(image, (1/downscale_factor, 1/downscale_factor, 1),

#converting image to grayscale
image_gray = color.rgb2gray(downscaled_image)

#apply gaussian blur to image (reducing noise)
blur_image = filters.gaussian(image_gray, sigma=1.0)

#calculate the gradient of the image using Sobel operator

sobel_x = filters.sobel(blur_image, axis=1)  #Gradient is the rate of change of ima
sobel_y = filters.sobel(blur_image, axis=0)

edges = feature.canny(blur_image, sigma=1.0, low_threshold=0.1, high_threshold=0.1)

#Display original iamge
plt.imshow(image)
plt.title('Original Image')
plt.axis('off')
plt.show()

#display image with edges
plt.imshow(edges, cmap='gray')
plt.title('Edge Detection')
plt.axis('off')
plt.show()
```



In [40]:
```
#increase value of sigma to 1.0 to 2.0 at threshold 0.1
```

In [41]:
```
import numpy as np
import matplotlib.pyplot as plt
from skimage import io, color, filters, feature
from scipy import ndimage
```

```
#for loading the image
image = io.imread('ct1.jpg')

downscale_factor = 2

# Perform datapreprocessing degrading resolution of picture
downscaled_image = ndimage.zoom(image, (1/downscale_factor, 1/downscale_factor, 1),

#converting image to grayscale
image_gray = color.rgb2gray(downscaled_image)

#apply gaussian blur to image (reducing noise)
blur_image = filters.gaussian(image_gray, sigma=1.0)

#calculate the gradient of the image using Sobel operator

sobel_x = filters.sobel(blur_image, axis=1)  #Gradient is the rate of change of ima
sobel_y = filters.sobel(blur_image, axis=0)

edges = feature.canny(blur_image, sigma=2.0, low_threshold=0.1, high_threshold=0.1)

#Display original iamge
plt.imshow(image)
plt.title('Original Image')
plt.axis('off')
plt.show()

#display image with edges
plt.imshow(edges, cmap='gray')
plt.title('Edge Detection')
plt.axis('off')
plt.show()
```

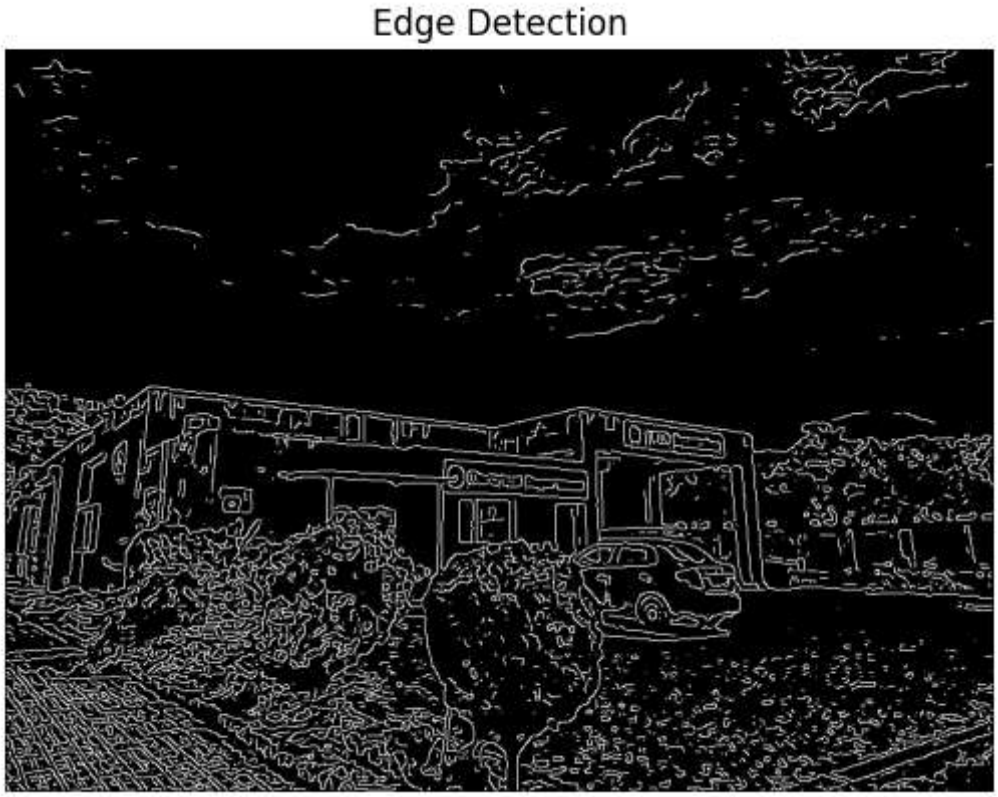


In [47]:
```
import numpy as np
import matplotlib.pyplot as plt
from skimage import io, color, filters, feature
from scipy import ndimage
```

```
#for loading the image
image = io.imread('ct2.jpg')

downscale_factor = 2

# Perform datapreprocessing degrading resolution of picture
downscaled_image = ndimage.zoom(image, (1/downscale_factor, 1/downscale_factor, 1),

#converting image to grayscale
image_gray = color.rgb2gray(downscaled_image)

#apply gaussian blur to image (reducing noise)
blur_image = filters.gaussian(image_gray, sigma=1.0)

#calculate the gradient of the image using Sobel operator

sobel_x = filters.sobel(blur_image, axis=1)  #Gradient is the rate of change of ima
sobel_y = filters.sobel(blur_image, axis=0)

edges = feature.canny(blur_image, sigma=2.0, low_threshold=0.1, high_threshold=0.1)

#Display original iamge
plt.imshow(image)
plt.title('Original Image')
plt.axis('off')
plt.show()

#display image with edges
plt.imshow(edges, cmap='gray')
plt.title('Edge Detection')
plt.axis('off')
plt.show()
```

In [43]:
```
import numpy as np
import matplotlib.pyplot as plt
from skimage import io, color, filters, feature
from scipy import ndimage
```

```
#for loading the image
image = io.imread('ct3.jpg')

downscale_factor = 2

# Perform datapreprocessing degrading resolution of picture
downscaled_image = ndimage.zoom(image, (1/downscale_factor, 1/downscale_factor, 1),

#converting image to grayscale
image_gray = color.rgb2gray(downscaled_image)

#apply gaussian blur to image (reducing noise)
blur_image = filters.gaussian(image_gray, sigma=1.0)

#calculate the gradient of the image using Sobel operator

sobel_x = filters.sobel(blur_image, axis=1)  #Gradient is the rate of change of ima
sobel_y = filters.sobel(blur_image, axis=0)

edges = feature.canny(blur_image, sigma=1.0, low_threshold=0.1, high_threshold=0.1)

#Display original iamge
plt.imshow(image)
plt.title('Original Image')
plt.axis('off')
plt.show()

#display image with edges
plt.imshow(edges, cmap='gray')
plt.title('Edge Detection')
plt.axis('off')
plt.show()
```

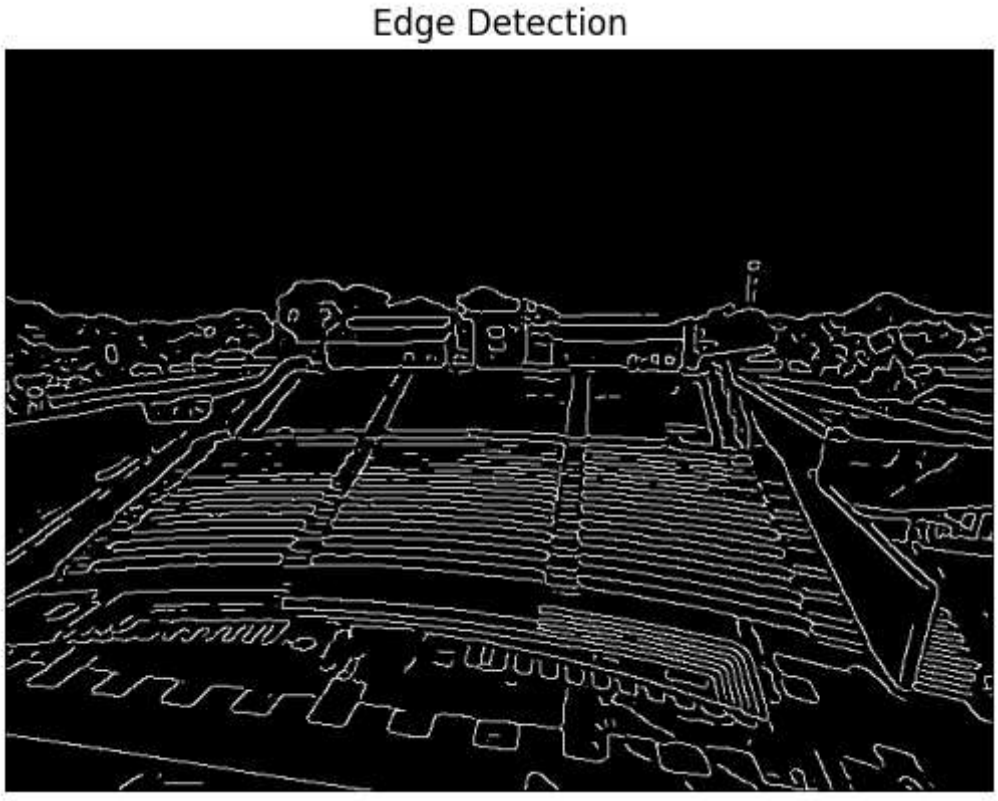


In [48]:
```
import numpy as np
import matplotlib.pyplot as plt
from skimage import io, color, filters, feature
from scipy import ndimage
```

```
#for loading the image
image = io.imread('ct4.jpg')

downscale_factor = 2

# Perform datapreprocessing degrading resolution of picture
downscaled_image = ndimage.zoom(image, (1/downscale_factor, 1/downscale_factor, 1),

#converting image to grayscale
image_gray = color.rgb2gray(downscaled_image)

#apply gaussian blur to image (reducing noise)
blur_image = filters.gaussian(image_gray, sigma=1.0)

#calculate the gradient of the image using Sobel operator

sobel_x = filters.sobel(blur_image, axis=1)  #Gradient is the rate of change of ima
sobel_y = filters.sobel(blur_image, axis=0)

edges = feature.canny(blur_image, sigma=2.0, low_threshold=0.1, high_threshold=0.1)

#Display original iamge
plt.imshow(image)
plt.title('Original Image')
plt.axis('off')
plt.show()

#display image with edges
plt.imshow(edges, cmap='gray')
plt.title('Edge Detection')
plt.axis('off')
plt.show()
```

In [49]:
```
import numpy as np
import matplotlib.pyplot as plt
from skimage import io, color, filters, feature
from scipy import ndimage
```

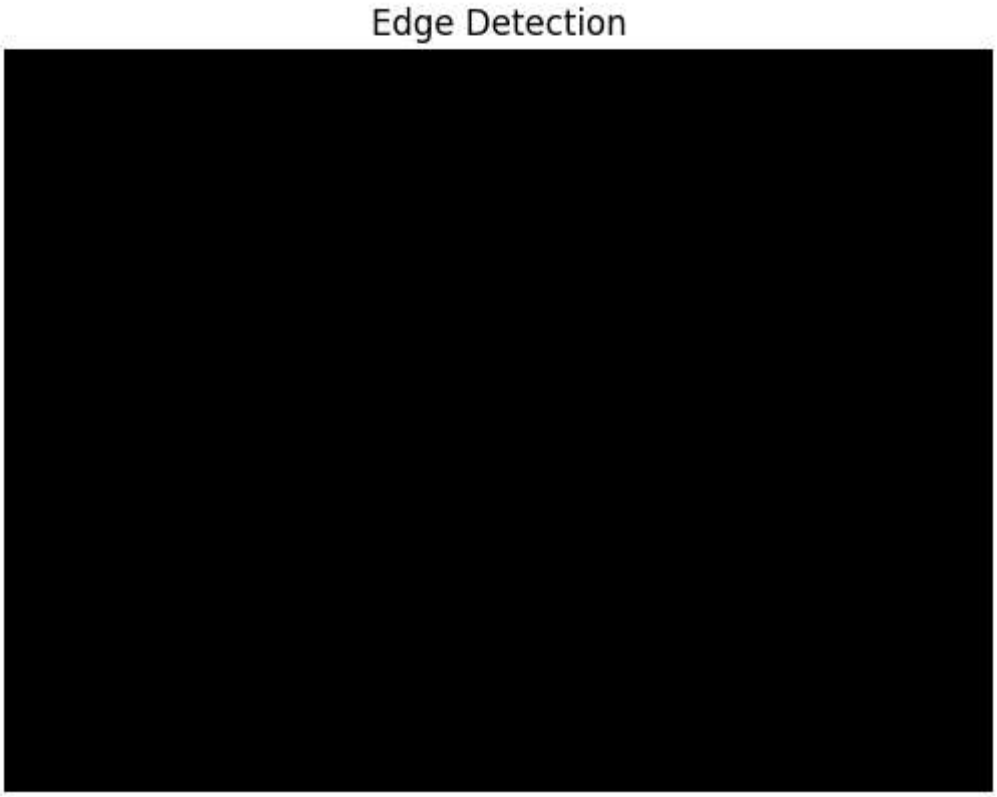
```
#for loading the image
image = io.imread('ct5.jpg')

downscale_factor = 2

# Perform datapreprocessing degrading resolution of picture
downscaled_image = ndimage.zoom(image, (1/downscale_factor, 1/downscale_factor, 1),

#converting image to grayscale
image_gray = color.rgb2gray(downscaled_image)

#apply gaussian blur to image (reducing noise)
blur_image = filters.gaussian(image_gray, sigma=1.0)

#calculate the gradient of the image using Sobel operator

sobel_x = filters.sobel(blur_image, axis=1)  #Gradient is the rate of change of ima
sobel_y = filters.sobel(blur_image, axis=0)

edges = feature.canny(blur_image, sigma=2.0, low_threshold=0.1, high_threshold=0.1)

#Display original iamge
plt.imshow(image)
plt.title('Original Image')
plt.axis('off')
plt.show()

#display image with edges
plt.imshow(edges, cmap='gray')
plt.title('Edge Detection')
plt.axis('off')
plt.show()
```



In [46]:
```
#Now try at higher threshold value with sigma = 2.0
```

In [50]:
```
import numpy as np
import matplotlib.pyplot as plt
from skimage import io, color, filters, feature
from scipy import ndimage
```

```
#for loading the image
image = io.imread('ct1.jpg')

downscale_factor = 2

# Perform datapreprocessing degrading resolution of picture
downscaled_image = ndimage.zoom(image, (1/downscale_factor, 1/downscale_factor, 1),

#converting image to grayscale
image_gray = color.rgb2gray(downscaled_image)

#apply gaussian blur to image (reducing noise)
blur_image = filters.gaussian(image_gray, sigma=1.0)

#calculate the gradient of the image using Sobel operator

sobel_x = filters.sobel(blur_image, axis=1)  #Gradient is the rate of change of ima
sobel_y = filters.sobel(blur_image, axis=0)

edges = feature.canny(blur_image, sigma=2.0, low_threshold=0.1, high_threshold=1.0)

#Display original iamge
plt.imshow(image)
plt.title('Original Image')
plt.axis('off')
plt.show()

#display image with edges
plt.imshow(edges, cmap='gray')
plt.title('Edge Detection')
plt.axis('off')
plt.show()
```

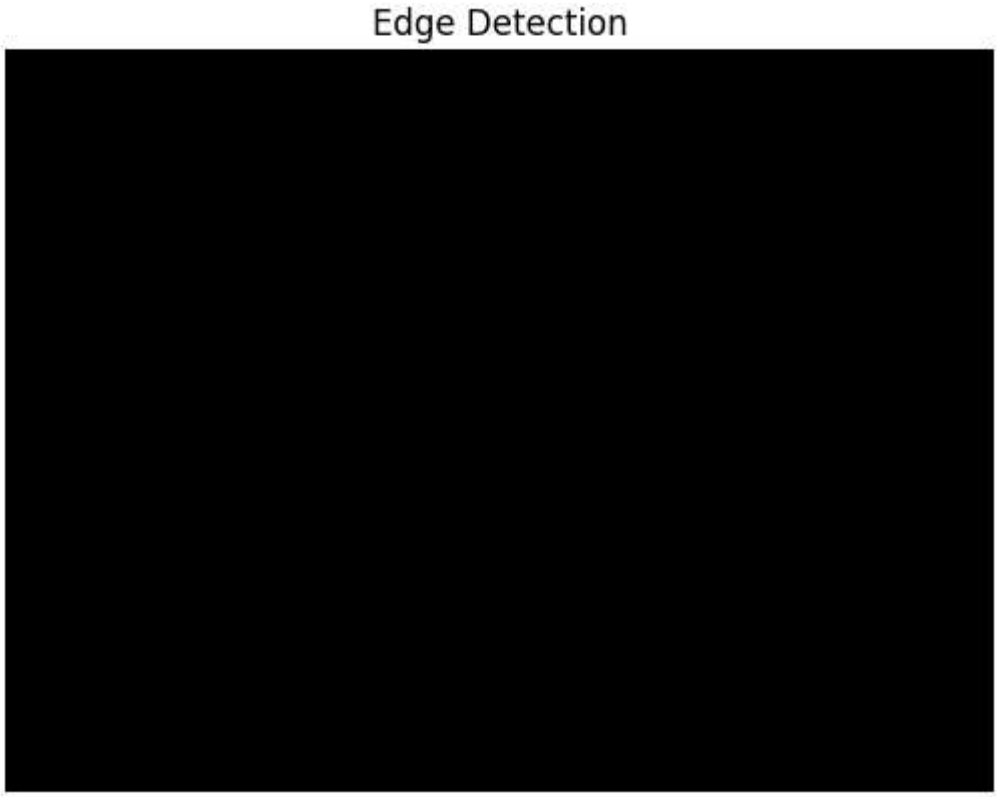


In [51]:
```
import numpy as np
import matplotlib.pyplot as plt
from skimage import io, color, filters, feature
from scipy import ndimage
```

```
#for loading the image
image = io.imread('ct2.jpg')

downscale_factor = 2

# Perform datapreprocessing degrading resolution of picture
downscaled_image = ndimage.zoom(image, (1/downscale_factor, 1/downscale_factor, 1),

#converting image to grayscale
image_gray = color.rgb2gray(downscaled_image)

#apply gaussian blur to image (reducing noise)
blur_image = filters.gaussian(image_gray, sigma=1.0)

#calculate the gradient of the image using Sobel operator

sobel_x = filters.sobel(blur_image, axis=1)  #Gradient is the rate of change of ima
sobel_y = filters.sobel(blur_image, axis=0)

edges = feature.canny(blur_image, sigma=2.0, low_threshold=0.1, high_threshold=1.0)

#Display original iamge
plt.imshow(image)
plt.title('Original Image')
plt.axis('off')
plt.show()

#display image with edges
plt.imshow(edges, cmap='gray')
plt.title('Edge Detection')
plt.axis('off')
plt.show()
```

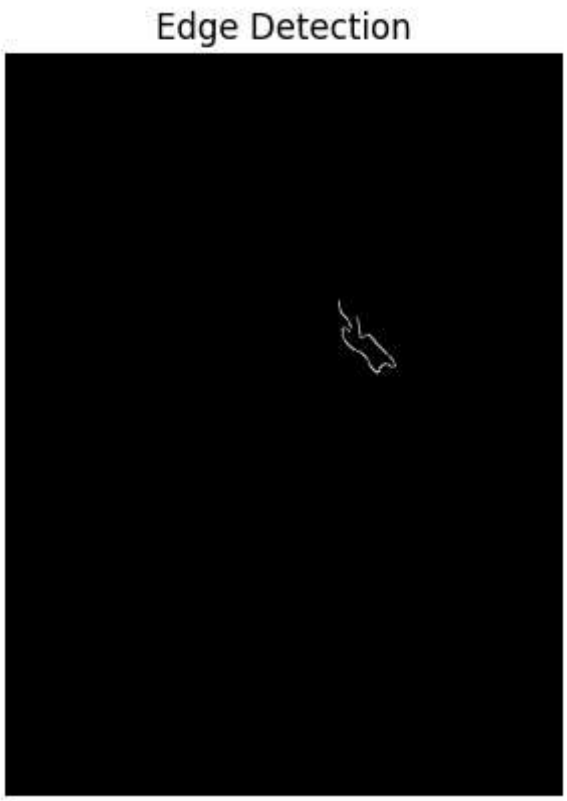
In [52]:
```
import numpy as np
import matplotlib.pyplot as plt
from skimage import io, color, filters, feature
from scipy import ndimage
```

```
#for loading the image
image = io.imread('ct3.jpg')

downscale_factor = 2

# Perform datapreprocessing degrading resolution of picture
downscaled_image = ndimage.zoom(image, (1/downscale_factor, 1/downscale_factor, 1),

#converting image to grayscale
image_gray = color.rgb2gray(downscaled_image)

#apply gaussian blur to image (reducing noise)
blur_image = filters.gaussian(image_gray, sigma=1.0)

#calculate the gradient of the image using Sobel operator

sobel_x = filters.sobel(blur_image, axis=1)  #Gradient is the rate of change of ima
sobel_y = filters.sobel(blur_image, axis=0)

edges = feature.canny(blur_image, sigma=2.0, low_threshold=0.1, high_threshold=1.0)

#Display original iamge
plt.imshow(image)
plt.title('Original Image')
plt.axis('off')
plt.show()

#display image with edges
plt.imshow(edges, cmap='gray')
plt.title('Edge Detection')
plt.axis('off')
plt.show()
```



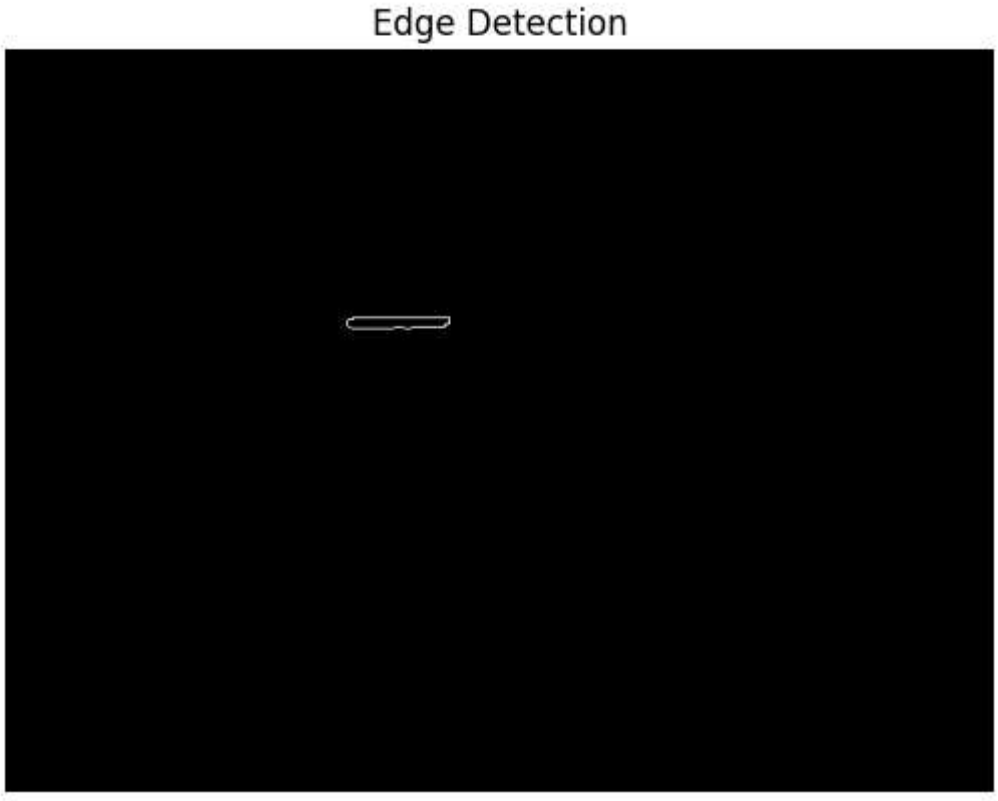
In [53]:
```
import numpy as np
import matplotlib.pyplot as plt
from skimage import io, color, filters, feature
from scipy import ndimage
```

```
#for loading the image
image = io.imread('ct4.jpg')

downscale_factor = 2

# Perform datapreprocessing degrading resolution of picture
downscaled_image = ndimage.zoom(image, (1/downscale_factor, 1/downscale_factor, 1),

#converting image to grayscale
image_gray = color.rgb2gray(downscaled_image)

#apply gaussian blur to image (reducing noise)
blur_image = filters.gaussian(image_gray, sigma=1.0)

#calculate the gradient of the image using Sobel operator

sobel_x = filters.sobel(blur_image, axis=1)  #Gradient is the rate of change of ima
sobel_y = filters.sobel(blur_image, axis=0)

edges = feature.canny(blur_image, sigma=2.0, low_threshold=0.1, high_threshold=1.0)

#Display original iamge
plt.imshow(image)
plt.title('Original Image')
plt.axis('off')
plt.show()

#display image with edges
plt.imshow(edges, cmap='gray')
plt.title('Edge Detection')
plt.axis('off')
plt.show()
```



In [54]:
```
import numpy as np
import matplotlib.pyplot as plt
from skimage import io, color, filters, feature
from scipy import ndimage
```

```
#for loading the image
image = io.imread('ct5.jpg')

downscale_factor = 2

# Perform datapreprocessing degrading resolution of picture
downscaled_image = ndimage.zoom(image, (1/downscale_factor, 1/downscale_factor, 1),

#converting image to grayscale
image_gray = color.rgb2gray(downscaled_image)

#apply gaussian blur to image (reducing noise)
blur_image = filters.gaussian(image_gray, sigma=1.0)

#calculate the gradient of the image using Sobel operator

sobel_x = filters.sobel(blur_image, axis=1)  #Gradient is the rate of change of ima
sobel_y = filters.sobel(blur_image, axis=0)

edges = feature.canny(blur_image, sigma=2.0, low_threshold=0.1, high_threshold=1.0)

#Display original iamge
plt.imshow(image)
plt.title('Original Image')
plt.axis('off')
plt.show()

#display image with edges
plt.imshow(edges, cmap='gray')
plt.title('Edge Detection')
plt.axis('off')
plt.show()
```



In [ ]: