Nisha Kini || Roll No: C050

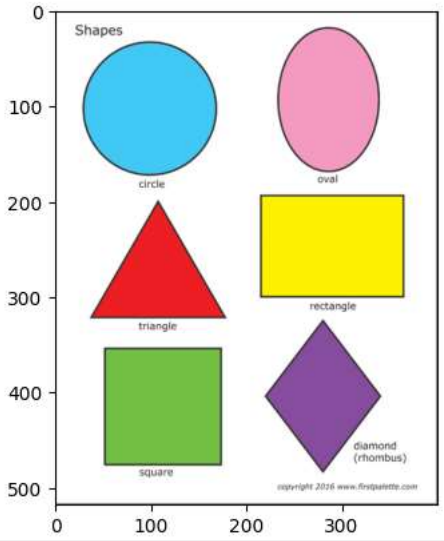
AIM: To detect edges using harris corner detector

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

```
image1 = cv2.imread('/content/shapes.png')
image= cv2.cvtColor(image1, cv2.COLOR_BGR2RGB)
```

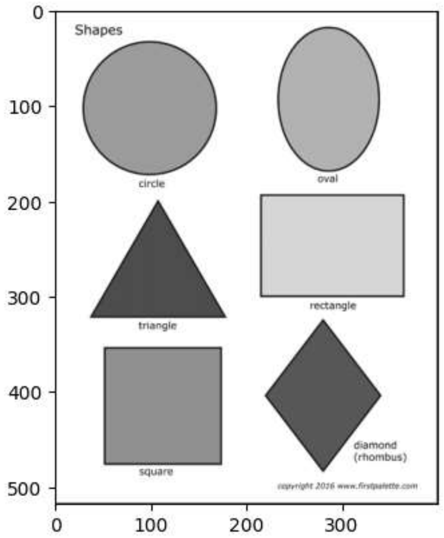
```
plt.imshow(image)
```

```
<matplotlib.image.AxesImage at 0x78a27efdd2d0>
```



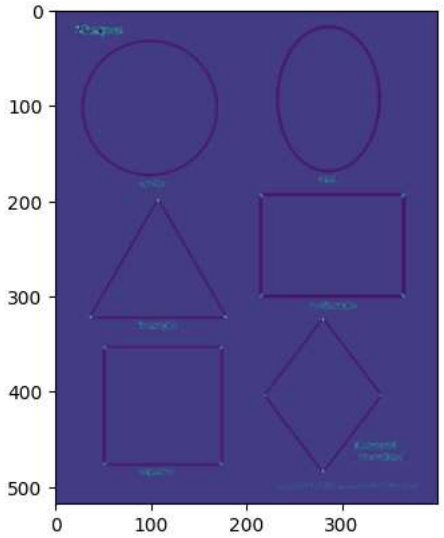
```
imag_g=cv2.cvtColor(image, cv2.COLOR_RGB2GRAY)
plt.imshow(imag_g, cmap='gray')
```

```
<matplotlib.image.AxesImage at 0x78a27f079c50>
```



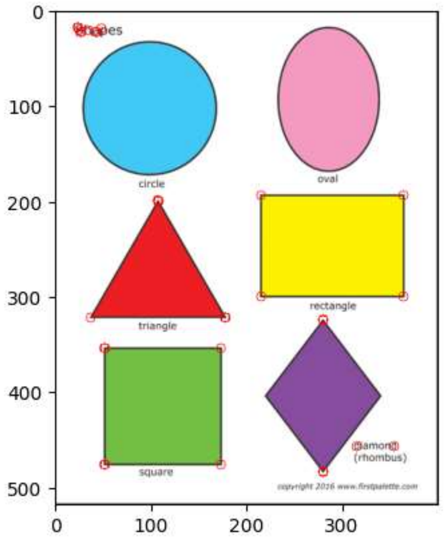
```
img_cr = cv2.cornerHarris(imag_g, 3, 3, 0.04)
plt.imshow(img_cr)
```

```
<matplotlib.image.AxesImage at 0x78a27eef5410>
```



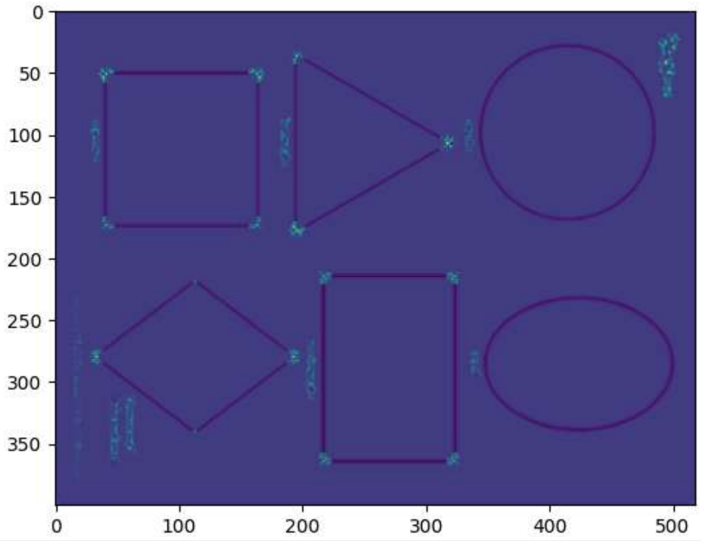
```
th=0.6*img_cr.max()
[rows,cols] = img_cr.shape
for r in range(0,rows):
  for c in range (0,cols):
    if img_cr[r,c]>th:
      cv2.circle(image, (c,r), 5, (255,0,0), 1)
    else:
      img_cr[r,c]=0
plt.imshow(image, cmap='gray')
```

```
<matplotlib.image.AxesImage at 0x78a27ef5d2d0>
```



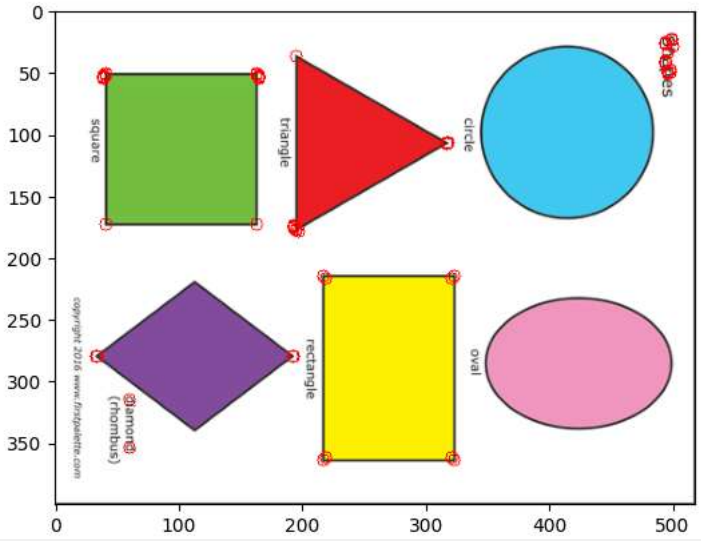
```
img = cv2.rotate(image, cv2.ROTATE_90_CLOCKWISE)
img_1g=cv2.cvtColor(img, cv2.COLOR_RGB2GRAY)
img_1g_cr= cv2.cornerHarris(img_1g, 3, 3, 0.04)
plt.imshow(img_1g_cr)
```

```
<matplotlib.image.AxesImage at 0x78a27edccf10>
```



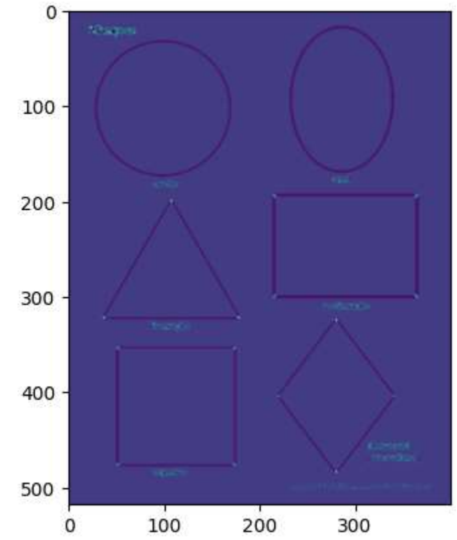
```
th=0.6*img_1g_cr.max()
[rows,cols] = img_1g_cr.shape
for r in range(0,rows):
  for c in range (0,cols):
    if img_1g_cr[r,c]>th:
      cv2.circle(img, (c,r), 5, (255,0,0), 1)
    else:
      img_1g_cr[r,c]=0
plt.imshow(img, cmap='gray')
```

```
<matplotlib.image.AxesImage at 0x78a27ee38250>
```



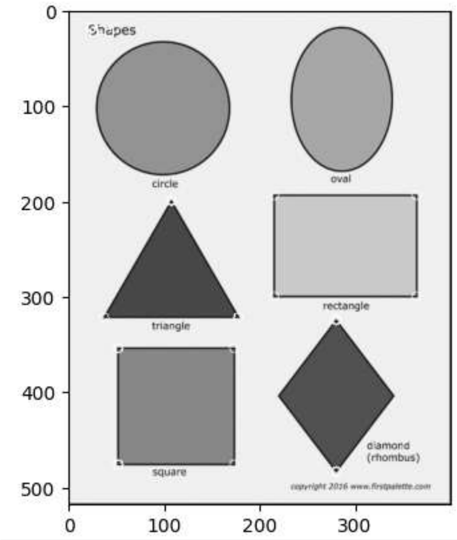
```
img2 = cv2.convertScaleAbs(imag_g, beta=-15)
img_2g_cr= cv2.cornerHarris(img2, 3, 3, 0.04)
plt.imshow(img_2g_cr)
```

```
<matplotlib.image.AxesImage at 0x78a27eea9450>
```



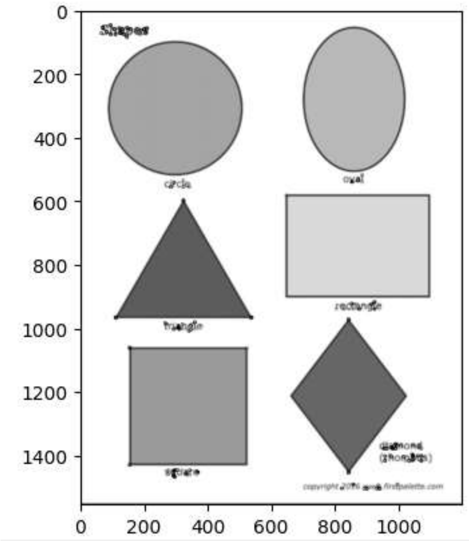
```
th=0.6*img_2g_cr.max()
[rows,cols] = img_2g_cr.shape
for r in range(0,rows):
  for c in range (0,cols):
    if img_2g_cr[r,c]>th:
      cv2.circle(img2, (c,r), 5, (255,0,0), 1)
    else:
      img_2g_cr[r,c]=0
plt.imshow(img2, cmap='gray')
```

```
<matplotlib.image.AxesImage at 0x78a27ed21990>
```



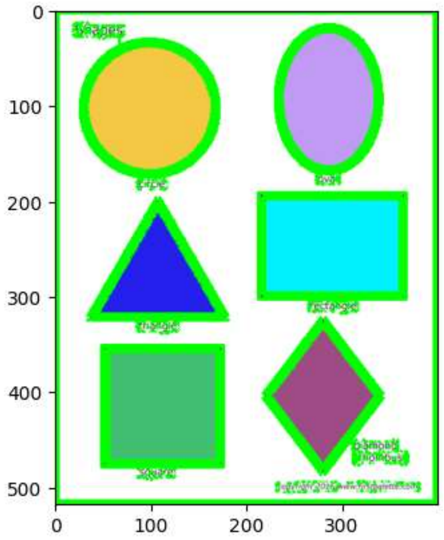
```
img3= cv2.resize(imag_g,(cols*3,rows*3))
img_3g_cr= cv2.cornerHarris(img3, 5, 5, 0.04)
th=0.4*img_3g_cr.max()
[rows,cols] = img_3g_cr.shape
for r in range(0,rows):
  for c in range (0,cols):
    if img_3g_cr[r,c]>th:
      cv2.circle(img3, (c,r), 5, (0,255,0), 1)
    else:
      img_3g_cr[r,c]=0
plt.imshow(img3, cmap='gray')
```

```
<matplotlib.image.AxesImage at 0x78a27ed87d10>
```



```
th_neg = 0.6*img_cr.min()
img_1g_cr= cv2.cornerHarris(imag_g, 3, 3, 0.04)
[rows,cols] = img_1g_cr.shape
for r in range(0,rows):
  for c in range (0,cols):
    if img_1g_cr[r,c]<th_neg:
      cv2.circle(image1, (c,r), 1, (0,255,0), 1)
    else:
      img_1g_cr[r,c]=0
plt.imshow(image1, cmap='gray')
```

```
<matplotlib.image.AxesImage at 0x78a27ebfb390>
```



```
fig = plt.figure(figsize=(10,10))
plt.subplot(2,2,1)
plt.title('Orignal image')
plt.imshow(image, cmap="gray")
plt.axis('off')
plt.subplot(2,2,2)
plt.title('image  with beta correction')
plt.imshow(img2, cmap='gray')

plt.axis('off')
plt.subplot(2,2,3)
plt.title('image  sized')
plt.imshow(img3, cmap='gray')
plt.axis('off')

plt.subplot(2,2,4)
plt.title('image  with negative threshold')
plt.imshow(image1, cmap='gray')
plt.axis('off')
plt.show()
```

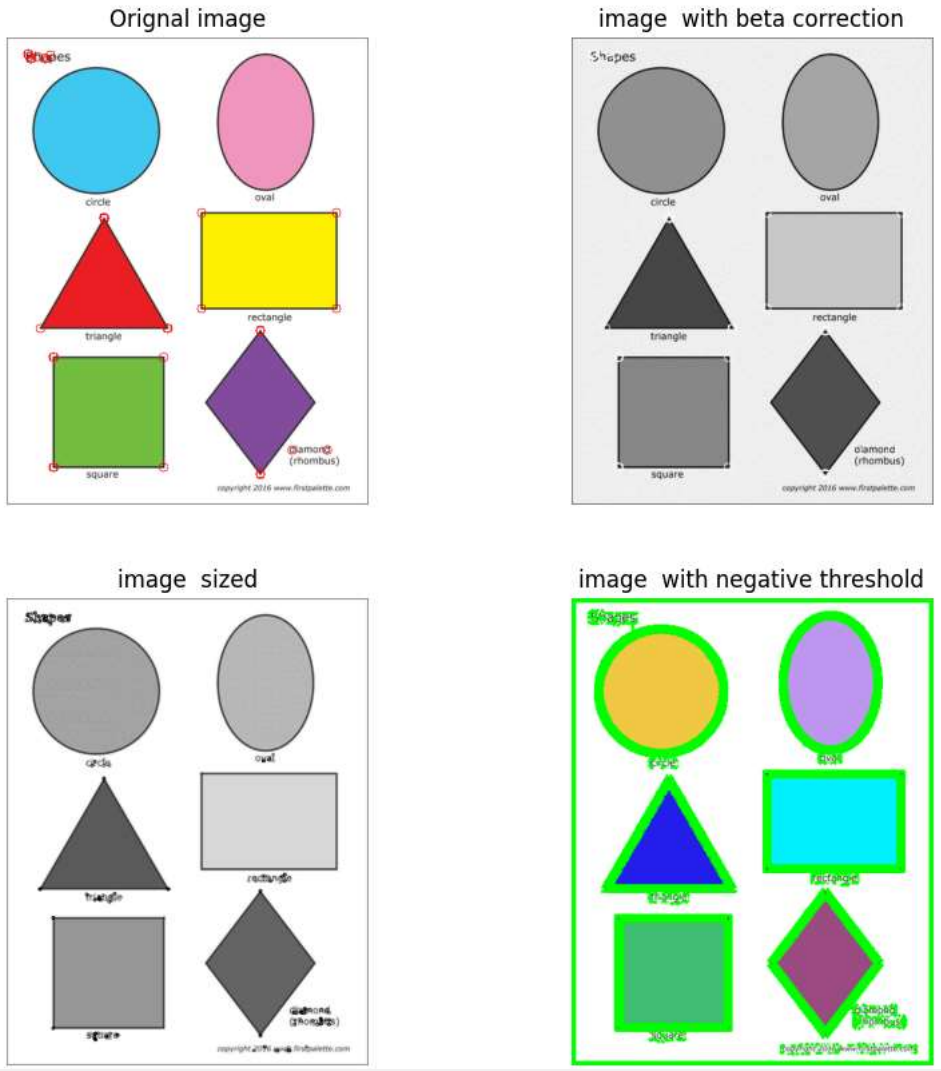

Orignal image

image with beta correction

image sized

image with negative threshold

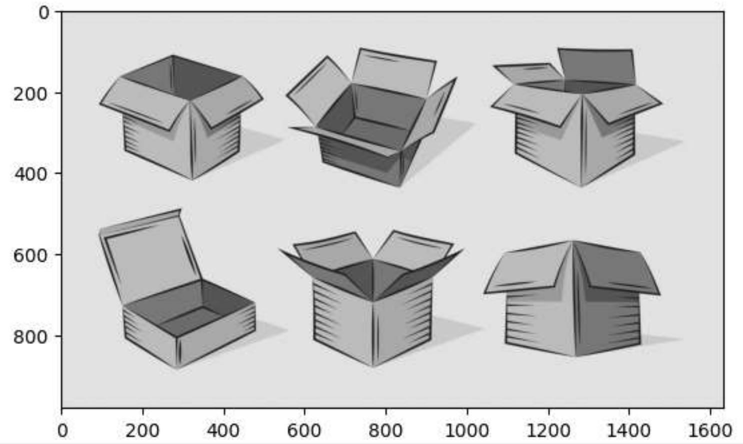
Conclusion: Harris corner detector is used to detect corners of given image, shape.png, for the threshold of if corner response is more than 60% of the maximum positive value, detector detects all the corners of the given image, including small text.

If the image is rotated by 90 deg or or brightness is reduced, it detects all the corners, if size of the image is increased to 3 times its og size,then most of the corners aren't detected by the detector, due to corners being shown as edges for the larger image, To avoid this issue, size of the window should be increased.

To detect edges, the threshold is considered as negative, which is equal to 60% of min value of corner response.

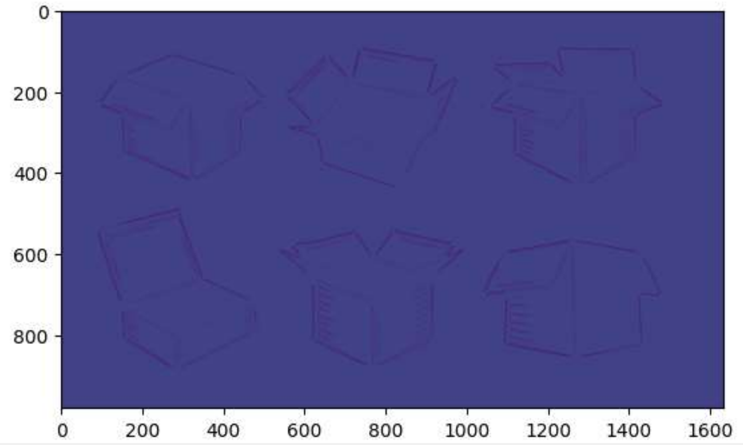
```
imageem = cv2.imread('/content/collection-of-flat-illustrations-of-cardboard-boxes-in-cartoon-style-perfect-for-illustrations-of-shipping-services-cargo-and-gift-boxes-free-vector.jpg')
imagem= cv2.cvtColor(imageem, cv2.COLOR_BGR2RGB)


imag_gm=cv2.cvtColor(imagem, cv2.COLOR_RGB2GRAY)
plt.imshow(imag_gm, cmap='gray')
```

```
<matplotlib.image.AxesImage at 0x78a27cfe3550>
```



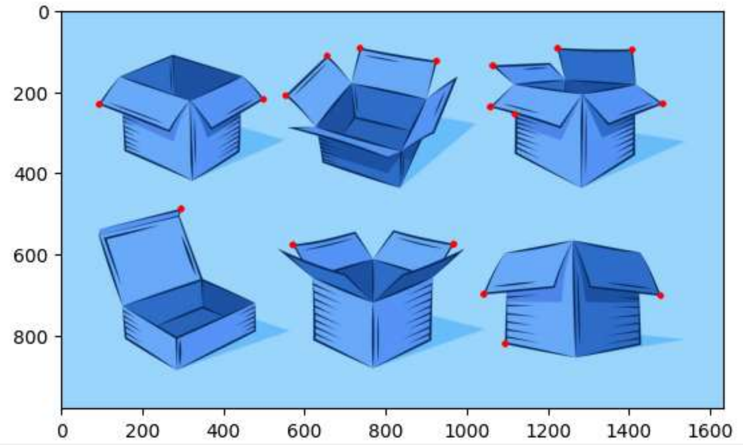
```
img_crm = cv2.cornerHarris(imag_gm, 3, 3, 0.04)
plt.imshow(img_crm)
```

```
<matplotlib.image.AxesImage at 0x78a27d05be50>
```



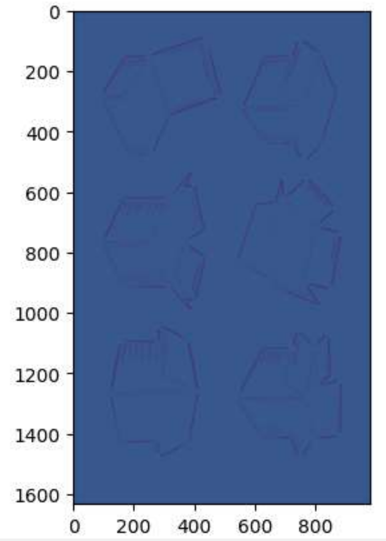
```
th=0.6*img_crm.max()
[rows,cols] = img_crm.shape
for r in range(0,rows):
  for c in range (0,cols):
    if img_crm[r,c]>th:
      cv2.circle(imageem, (c,r), 5, (255,0,0), 5)
    else:
      img_crm[r,c]=0
plt.imshow(imageem, cmap='gray')
```

```
<matplotlib.image.AxesImage at 0x78a27ced9710>
```



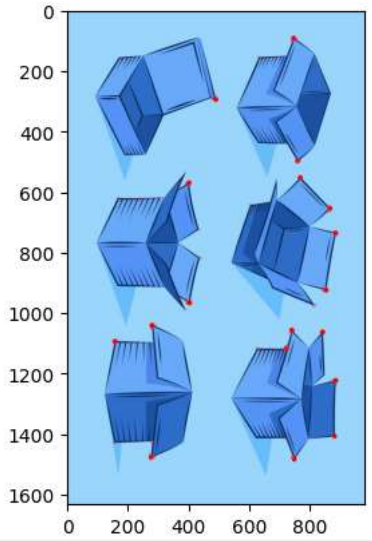
```
imgs = cv2.rotate(imageem, cv2.ROTATE_90_CLOCKWISE)
img_1gs=cv2.cvtColor(imgs, cv2.COLOR_RGB2GRAY)
img_1gs_cr= cv2.cornerHarris(img_1gs, 3, 3, 0.04)
plt.imshow(img_1gs_cr)
```

```
<matplotlib.image.AxesImage at 0x78a27cf4b490>
```



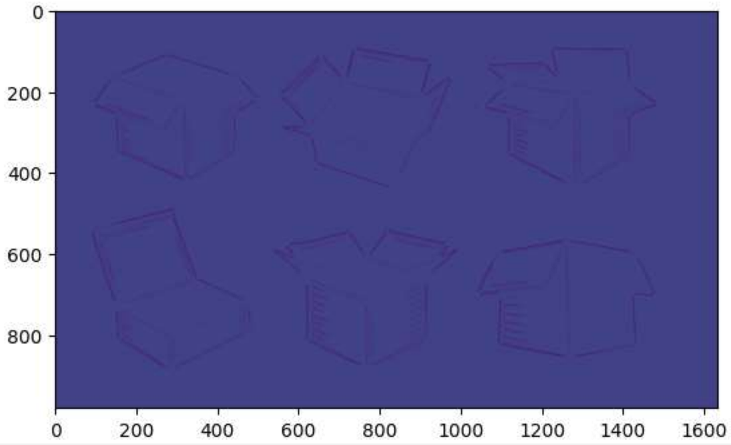
```
th=0.6*img_1gs_cr.max()
[rows,cols] = img_1gs_cr.shape
for r in range(0,rows):
  for c in range (0,cols):
    if img_1gs_cr[r,c]>th:
      cv2.circle(imgs, (c,r), 5, (255,0,0), 1)
    else:
      img_1gs_cr[r,c]=0
plt.imshow(imgs, cmap='gray')
```

```
<matplotlib.image.AxesImage at 0x78a27cdca5d0>
```



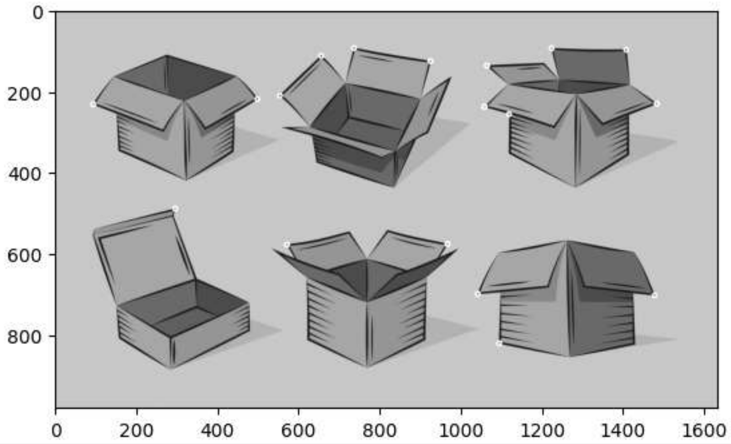
```
img2 = cv2.convertScaleAbs(imag_gm, beta=-15)
img_2g_cr= cv2.cornerHarris(img2, 3, 3, 0.04)
plt.imshow(img_2g_cr)
```

```
<matplotlib.image.AxesImage at 0x78a27ce0d2d0>
```



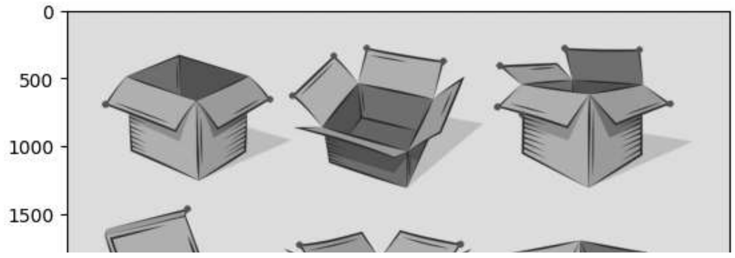
```
th=0.6*img_2g_cr.max()
[rows,cols] = img_2g_cr.shape
for r in range(0,rows):
  for c in range (0,cols):
    if img_2g_cr[r,c]>th:
      cv2.circle(img2, (c,r), 5, (255,0,0), 3)
    else:
      img_2g_cr[r,c]=0
plt.imshow(img2, cmap='gray')
```

```
<matplotlib.image.AxesImage at 0x78a27ccbea50>
```



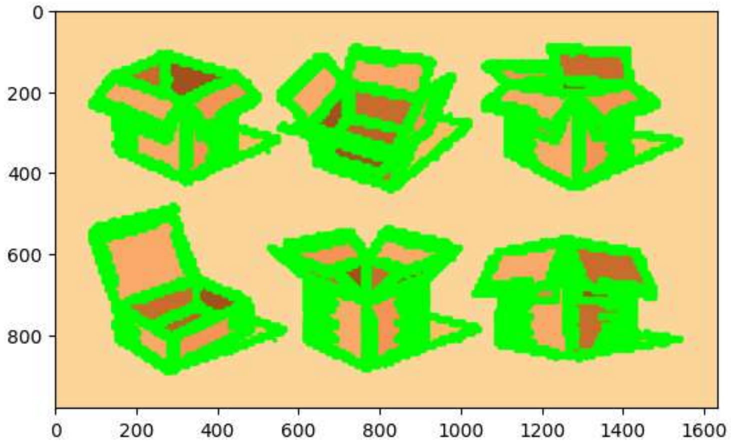
```
img3= cv2.resize(imageem,(cols*3,rows*3))
img_3=cv2.cvtColor(img3, cv2.COLOR_RGB2GRAY)
img_3g_cr= cv2.cornerHarris(img_3, 5, 5, 0.04)
th=0.6*img_3g_cr.max()
[rows,cols] = img_3g_cr.shape
for r in range(0,rows):
  for c in range (0,cols):
    if img_3g_cr[r,c]>th:
      cv2.circle(img_3,(c,r), 2, (0,255,0), 3)
    else:
      img_3g_cr[r,c]=0
plt.imshow(img_3, cmap='gray')
```

```
<matplotlib.image.AxesImage at 0x78a27cb30c90>
```



```
th_neg = 1.2*img_cr.min()
img_1gs_cr= cv2.cornerHarris(imag_gm, 1, 1, 0.04)
[rows,cols] = img_1gs_cr.shape
for r in range(0,rows):
  for c in range (0,cols):
    if img_1gs_cr[r,c]<th_neg:
      cv2.circle(imagem,(c,r), 1, (0,255,0), 7)
    else:
      img_1gs_cr[r,c]=0
plt.imshow(imagem, cmap='gray')
```

```
<matplotlib.image.AxesImage at 0x78a27cd94690>
```



```
fig = plt.figure(figsize=(10,5))
plt.subplot(2,2,1)
plt.title('Orignal image')
plt.imshow(imageem, cmap="gray")
plt.axis('off')
plt.subplot(2,2,2)
plt.title('image  with beta correction')
plt.imshow(img2, cmap='gray')

plt.axis('off')
plt.subplot(2,2,3)
plt.title('image  sized')
plt.imshow(img_3, cmap='gray')
plt.axis('off')

plt.subplot(2,2,4)
plt.title('image  with negative threshold')
plt.imshow(imagem, cmap='gray')
plt.axis('off')
plt.show()
```

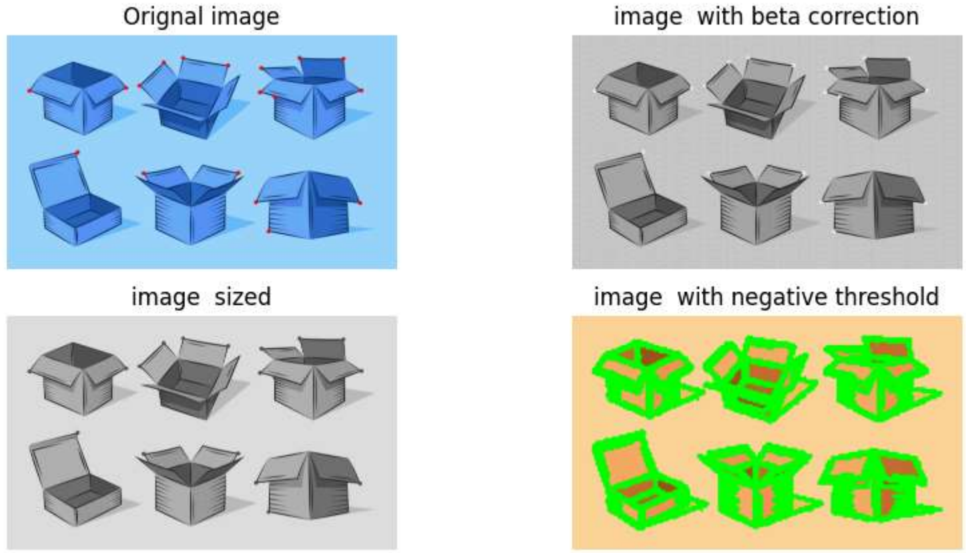

Orignal image

image with beta correction

image sized

image with negative threshold