Harris Corner Detector

Import necessary libraries

In [1]:
```
import numpy as np
import cv2
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
```

Sobel and gaussian kernels defined here with numpy

In [2]:
```
#Defining Sobel Kernels X-axis
SOBEL_X = np.array((
    [-1, 0, 1],
    [-2, 0, 2],
    [-1, 0, 1]), dtype="int32")

# Sobel y-axis kernel
SOBEL_Y = np.array((
    [-1, -2, -1],
    [0, 0, 0],
    [1, 2, 1]), dtype="int32")

# Gaussian kernel
GAUSS = np.array((
    [1/16, 2/16, 1/16],
    [2/16, 4/16, 2/16],
    [1/16, 2/16, 1/16]), dtype="float64")
```

In [3]:
```
def convolve(image, kernel, average=False):

    image_row, image_col = image.shape
    kernel_row, kernel_col = kernel.shape

    output = np.zeros(image.shape)

    pad_height = int((kernel_row - 1) / 2)
    pad_width = int((kernel_col - 1) / 2)

    padded_image = np.zeros((image_row + (2 * pad_height), image_col + (2 * pad_

    padded_image[pad_height:padded_image.shape[0] - pad_height, pad_width:padded

    for row in range(image_row):
        for col in range(image_col):
            output[row, col] = np.sum(kernel * padded_image[row:row + kernel_row
            if average:
                output[row, col] /= kernel.shape[0] * kernel.shape[1]

    return output
```

Convolution function from scratch.

Alternatively, you can use opencv, skimage, pillow libraries.

Harris function

- Image converted to grayscale
- Sobel filters convolved on image for both axis
- Square of derivatives and cross multiply calculated
- All axes filtered with gaussian kernel
- Cornerness Function used to calculate
- Non-max supression implemented

In [4]:

```
def harris(img, threshold=0.6):
    # Copy Image
    img1_ = img.copy()

    #Step 1: Sobel Gradient horizontal and Vertical
    dx = convolve(img1_, SOBEL_X) # convolving with sobel filter on X-axis
    dy = convolve(img1_, SOBEL_Y) # convolving with sobel filter on Y-axis

    print('Step 1 output:', dx, dy)

    #Step 2:  square of derivatives
    dx2 = np.square(dx)
    dy2 = np.square(dy)
    dxdy = dx*dy
    print('Step 2 output:', dx2, dy2, dxdy)

    #Steps 3: Apply gauss filter for all directions (x,y,cross axis)
    g_dx2 = convolve(dx2, GAUSS)
    g_dy2 = convolve(dy2, GAUSS)
    g_dxdy = convolve(dxdy, GAUSS)
    print('Step 3 output:', g_dx2, g_dy2, g_dxdy)

    # Step4: Harris Function Corner Detector: r(harris) = det - k*(trace**2)
    harris = g_dx2*g_dy2 - np.square(g_dxdy) - 0.001*np.square(g_dx2 + g_dy2)
    # Normalizing inside (0-1)
    cv2.normalize(harris, harris, 0, 1, cv2.NORM_MINMAX)

    print('Step 4: Normalized Harris', harris)

    #Step 5: find all points above threshold (nonmax supression line)
    loc = np.where(harris >= threshold)
    print('Step 5: corner location')
    return loc
```

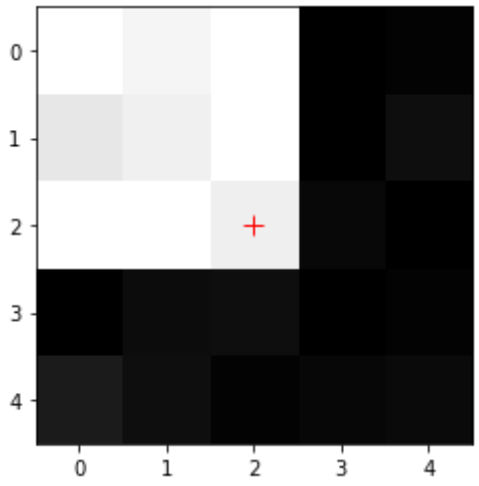
Image Reading and Filtering Area

In [6]:

```
img = mpimg.imread('corner.jpg')
```

```
loc = harris(img, 0.99)
fig, ax = plt.subplots()
ax.imshow(img, cmap=plt.cm.gray)
ax.plot(loc[1], loc[0]+1, color='red', marker='+',
        linestyle='None', markersize=10)
plt.show()
```

```
Step 1 output: [[ 730.   24. -730. -745.    0.]
 [ 980.   32. -972. -973.   -8.]
 [ 762.    6. -746. -730.  -16.]
 [ 293.  -12. -278. -254.  -15.]
 [  40.  -34.  -26.    3.  -14.]] [[ 702.  966.  750.  269.   28.]
 [  10.    4.  -14.   -3.    2.]
 [-690. -928. -710. -252.  -22.]
 [-697. -946. -714. -228.   19.]
 [ -12.  -38.  -40.  -17.   -6.]]
Step 2 output: [[5.32900e+05 5.76000e+02 5.32900e+05 5.55025e+05 0.00000e+00]
 [9.60400e+05 1.02400e+03 9.44784e+05 9.46729e+05 6.40000e+01]
 [5.80644e+05 3.60000e+01 5.56516e+05 5.32900e+05 2.56000e+02]
 [8.58490e+04 1.44000e+02 7.72840e+04 6.45160e+04 2.25000e+02]
 [1.60000e+03 1.15600e+03 6.76000e+02 9.00000e+00 1.96000e+02]] [[4.92804e+05 9.
33156e+05 5.62500e+05 7.23610e+04 7.84000e+02]
 [1.00000e+02 1.60000e+01 1.96000e+02 9.00000e+00 4.00000e+00]
 [4.76100e+05 8.61184e+05 5.04100e+05 6.35040e+04 4.84000e+02]
 [4.85809e+05 8.94916e+05 5.09796e+05 5.19840e+04 3.61000e+02]
 [1.44000e+02 1.44400e+03 1.60000e+03 2.89000e+02 3.60000e+01]] [[ 5.12460e+05
2.31840e+04 -5.47500e+05 -2.00405e+05  0.00000e+00]
 [ 9.80000e+03  1.28000e+02  1.36080e+04  2.91900e+03 -1.60000e+01]
 [-5.25780e+05 -5.56800e+03  5.29660e+05  1.83960e+05  3.52000e+02]
 [-2.04221e+05  1.13520e+04  1.98492e+05  5.79120e+04 -2.85000e+02]
 [-4.80000e+02  1.29200e+03  1.04000e+03 -5.10000e+01  8.40000e+01]]
Step 3 output: [[253411.     252571.     380007.6875 382762.875  128556.6875]
 [379459.25   376165.5    558875.6875 558883.375  186384.4375]
 [276019.625  271569.8125 396780.3125 393124.4375 129915.4375]
 [ 94335.25    91791.375  130433.8125 127284.      41484.0625]
 [ 11284.625   10787.3125  14016.375   13020.0625   4110.5   ]] [[239859.     36
5222.5    266340.6875  88514.375    9242.1875]
 [233286.25   351552.5    254014.9375  83752.125    8652.1875]
 [343344.875  511931.3125 364542.8125 117345.4375  11354.1875]
 [346761.5    517379.625  366925.3125 116378.5     10640.3125]
 [116874.875  174668.8125 123522.375   38659.5625   3339.25  ]] [[ 132246.
2895.     -157136.1875 -117324.375   -24870.1875]
 [   1902.        3212.5       1626.0625    -719.875     -622.9375]
 [-155726.125     1632.9375  185745.9375  133082.8125   26847.3125]
 [-115686.        1864.875   135845.5625   95438.5      18716.5625]
 [ -24776.625     1453.9375   29555.625    19754.6875    3598.5   ]]
Step 4: Normalized Harris [[3.04672962e-01 6.50068934e-01 5.38584466e-01 1.40781
167e-01
  3.89182132e-03]
 [6.23805327e-01 9.32077825e-01 1.00000000e+00 3.28336198e-01
  1.11358803e-02]
 [4.96359878e-01 9.79541289e-01 7.75389926e-01 1.99287326e-01
  5.19203708e-03]
 [1.35410346e-01 3.33447701e-01 2.06352786e-01 3.99474613e-02
  6.20355971e-04]
 [4.86814283e-03 1.30714505e-02 5.93185403e-03 7.76441942e-04
  0.00000000e+00]]
Step 5: corner location
```


0
1
2
3
4
0
1
2
3
4