Lab 07

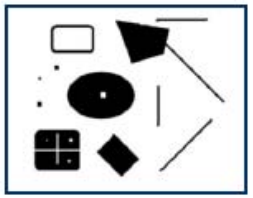
Harris Corner

Contents

- images
- derivatives
- Harris corner
- Find Peak

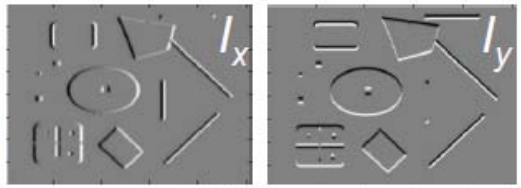
Harris Detector [Harris88]

❖ Second moment matrix

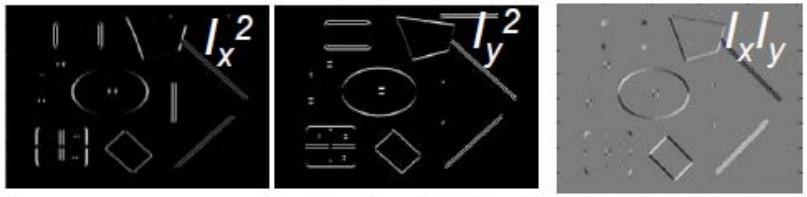


$$\mu(\sigma_I, \sigma_D) = g(\sigma_I) * \begin{bmatrix} I_x^2(\sigma_D) & I_x I_y(\sigma_D) \\ I_x I_y(\sigma_D) & I_y^2(\sigma_D) \end{bmatrix}$$

1. Image derivatives (optionally, blur first)



2. Square of derivatives



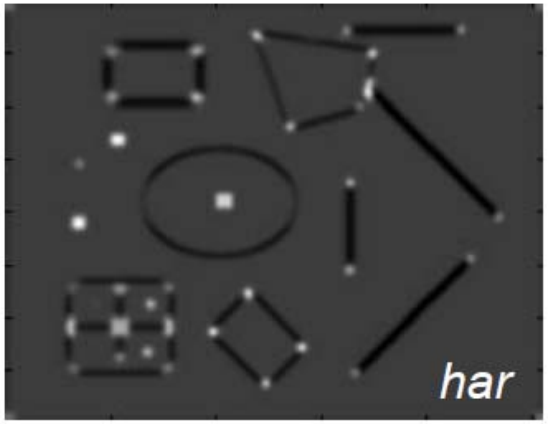
$$\det M = \lambda_1 \lambda_2$$
$$\text{trace}\, M = \lambda_1 + \lambda_2$$

3. Gaussian filter $g(\sigma_I)$

4. Cornerness function – both eigenvalues are strong

$$har = \det[\mu(\sigma_I, \sigma_D)] - \alpha[\text{trace}(\mu(\sigma_I, \sigma_D))^2] =$$
$$g(I_x^2)g(I_y^2) - [g(I_x I_y)]^2 - \alpha[g(I_x^2) + g(I_y^2)]^2$$

5. Non-maxima suppression



Corner response function

$$R = \lambda_1 \lambda_2 - \alpha(\lambda_1 + \lambda_2)^2 = \det(M) - \alpha \operatorname{trace}(M)^2$$

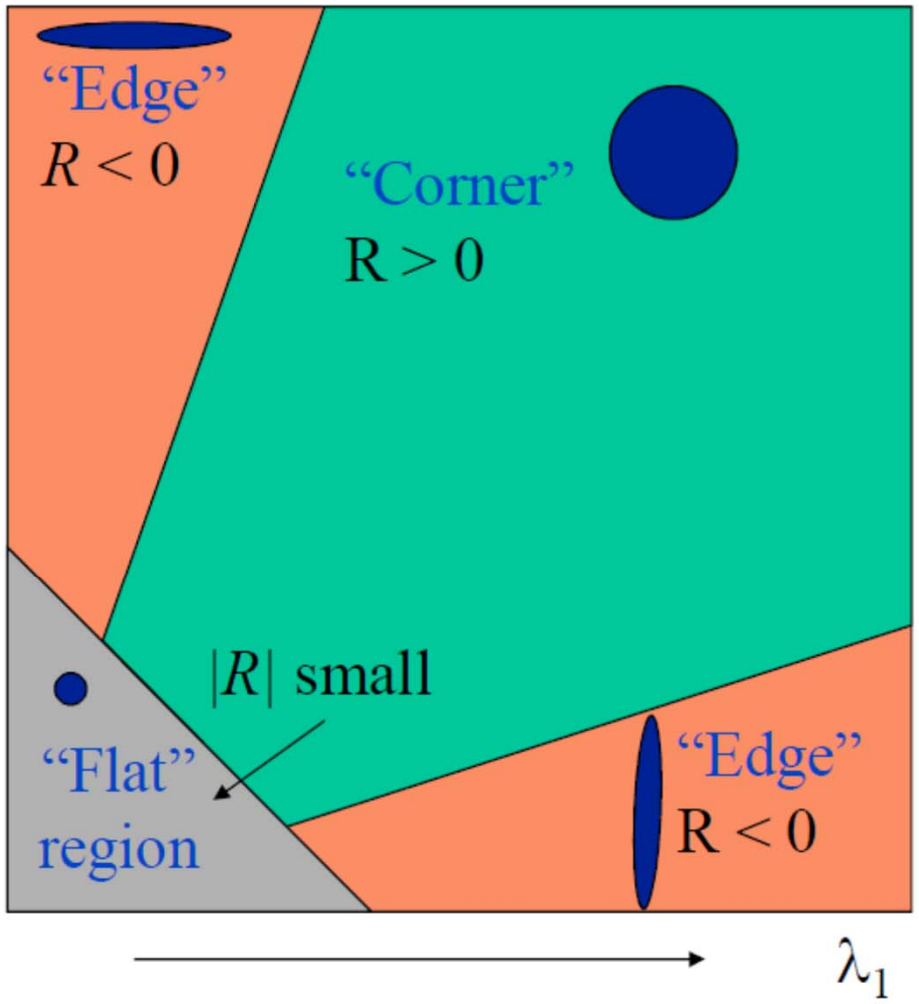
α: constant (0.04 to 0.06)

Determinant (det(A)):

$$|A| = \begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - bc.$$

Trace (trace(A)):

$$\operatorname{tr}(A) = a_{11} + a_{22} + \cdots + a_{nn} = \sum_{i=1}^{n} a_{ii}$$



images

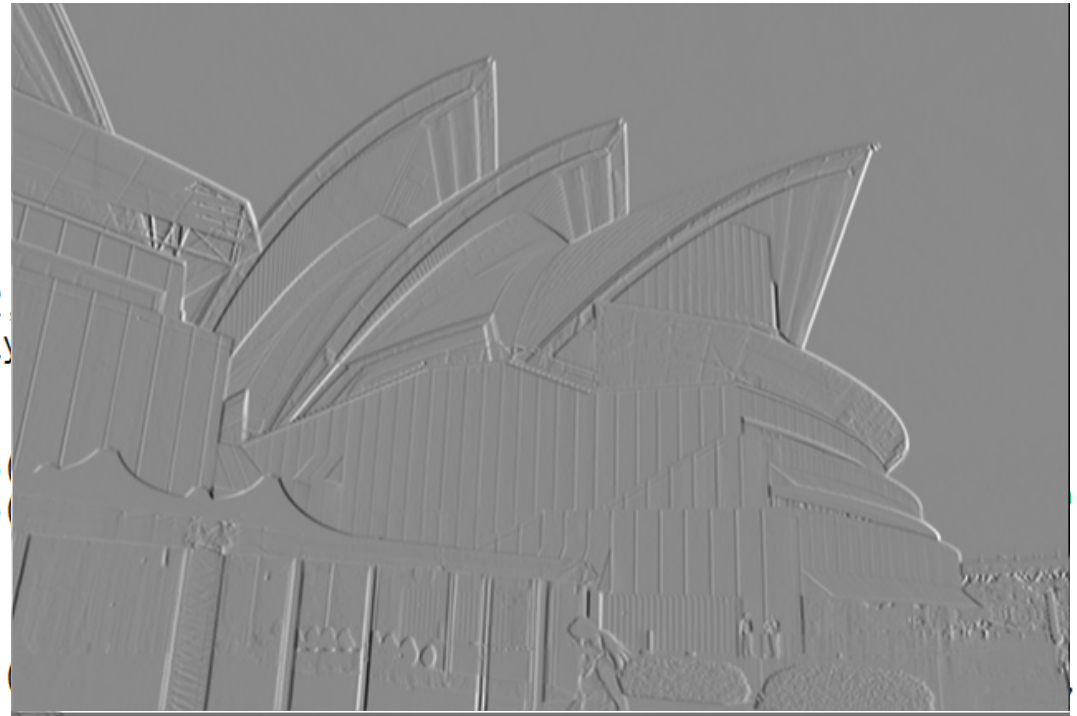
- Input image

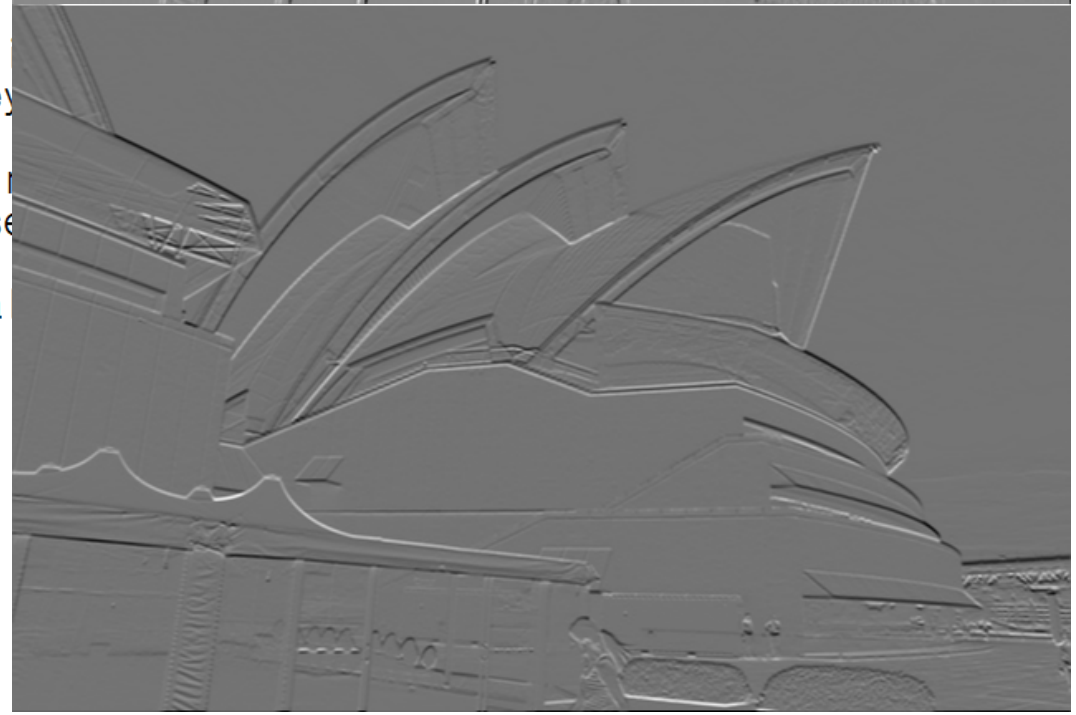
Expected output

main

```
img_input = io.imread( "sydney.jpg", as_grey=True )

img_response = harris( img_input )

idx = np.nonzero( img_response )

for (r,c) in zip(idx[0],idx[1]):
    rr,cc = circle(r,c, 5)
    try:
        img_input[rr,cc]=0
    except:
        pass

io.imsave( "harris.png", img_input )
```

```
def harris( image, sigma=2
    kernel_deriv = np.array


    img_deriv_x = convolve(
    img_deriv_y = convolve(                                    , cval=0 )

    igx = gaussian_filter(
    igy = gaussian_filter(
    igxy = gaussian_filter(                                    cval=0 )

    response = (igx*igy -
    response_dilated = grey

    img_max = response ==
    img_threshed = response

    img_corner = np.logica


    return img_corner
```

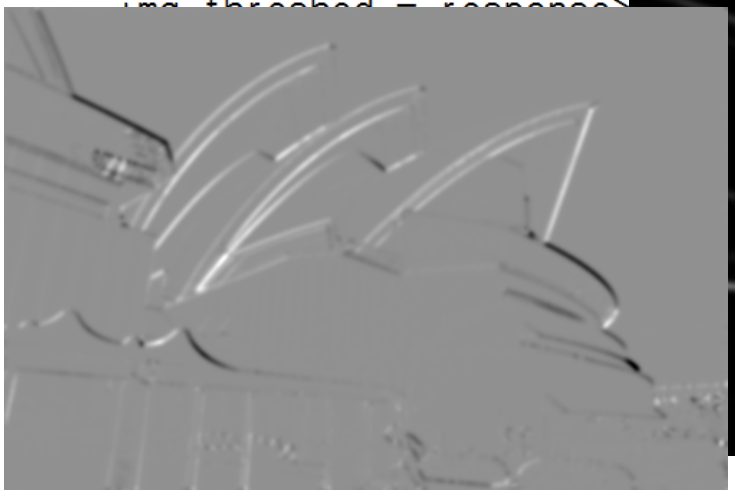


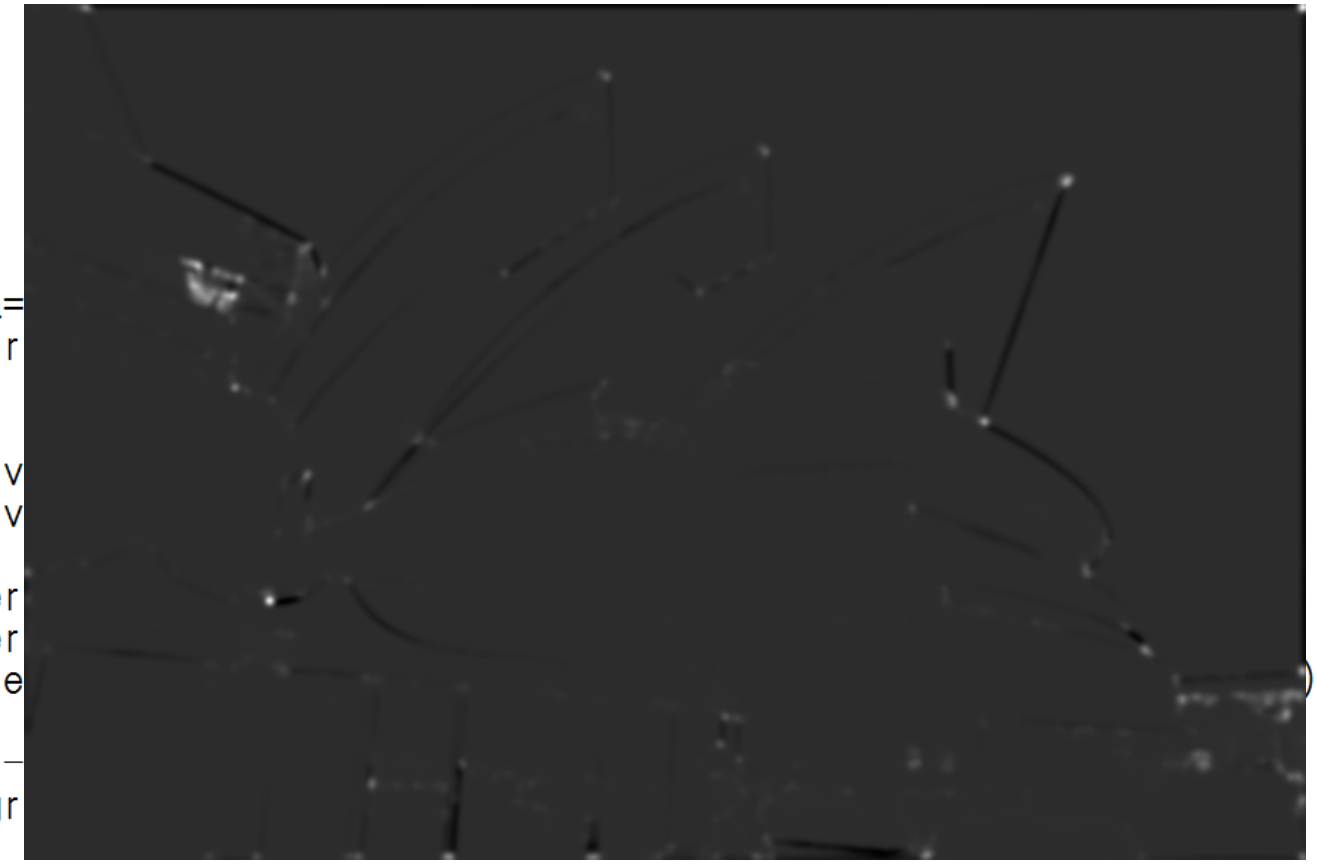


```
def harris( image, sigma=2,
    kernel_deriv = np.array(


    img_deriv_x = convolve(
    img_deriv_y = convolve(

    igx = gaussian_filter( i
    igy = gaussian_filter( i
    igxy = gaussian_filter(

    response = (igx*igy - ig
    response_dilated = grey_

    img_max = response == re
```

cval=0)

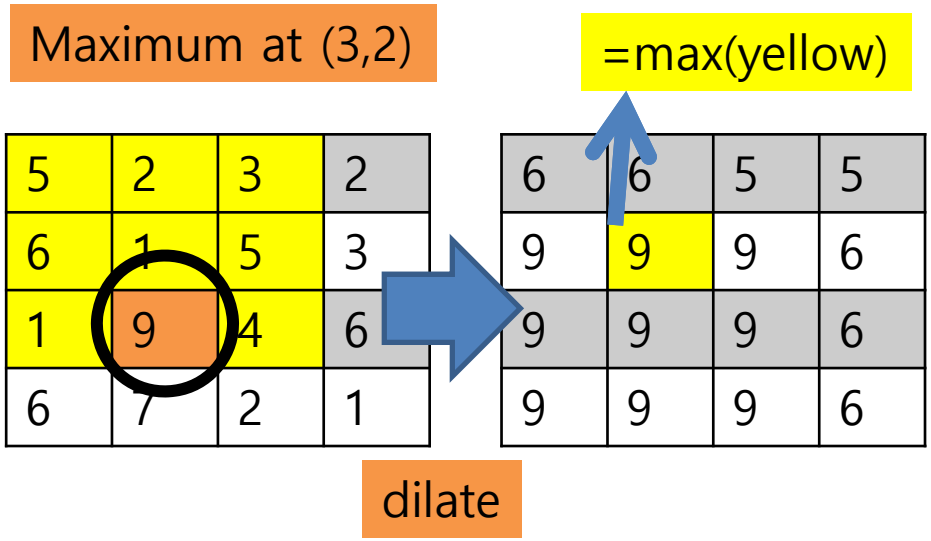
cval=0)

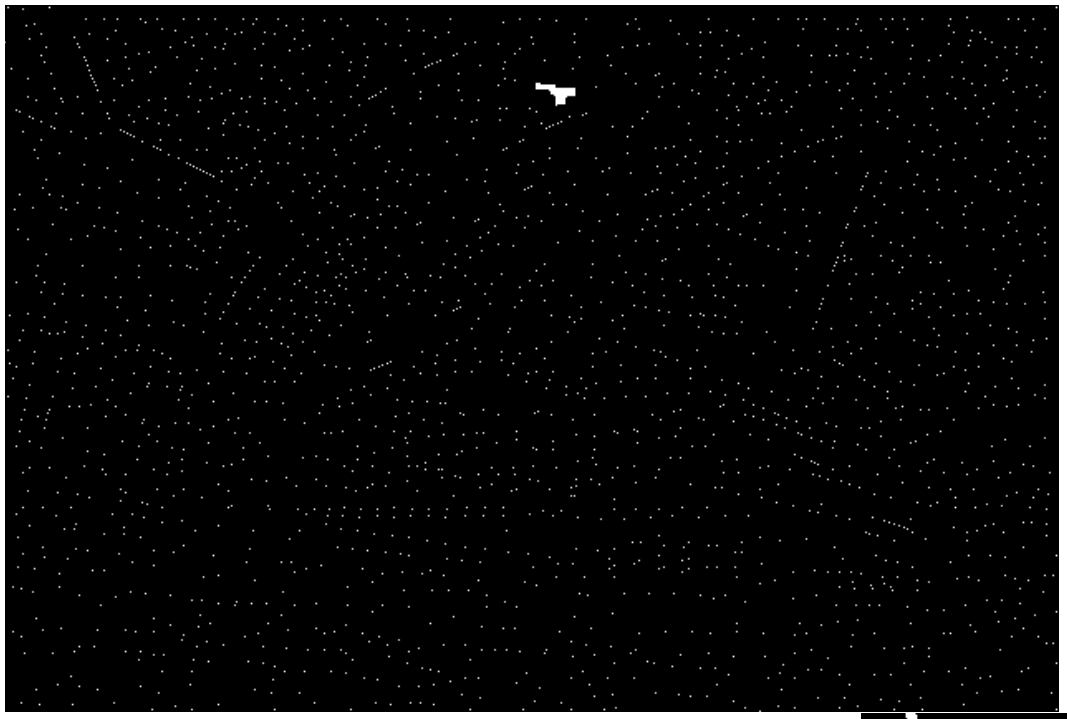


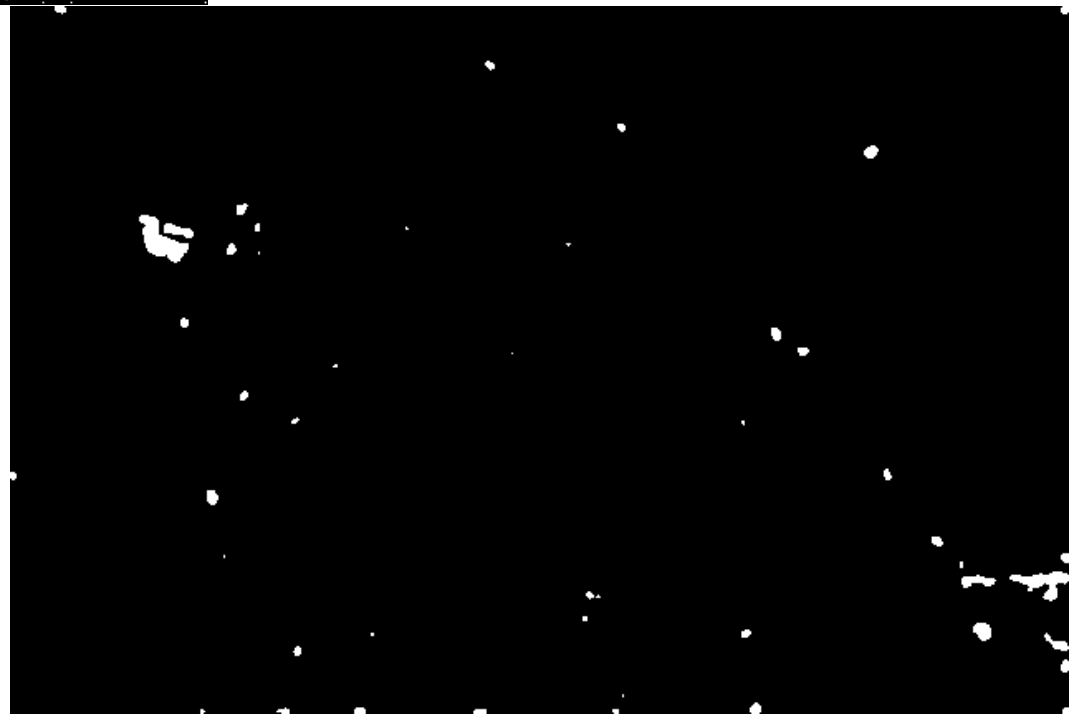


```
def harris( image, sigma=
    kernel_deriv = np.arr


    img_deriv_x = convolv
    img_deriv_y = convolv                                                    )

    igx = gaussian_filter
    igy = gaussian_filter
    igxy = gaussian_filte                                                    )

    response = (igx*igy -
    response_dilated = gr

    img_max = response == response_dilated
    img_threshed = response>thresh

    img_corner = np.logical_and(img_max,img_threshed)

    return img_corner
```

Find maximum by dilation







```
def harris( image, sigma=2, radius=3, alpha=0.04, thresh=0.2 ):
    kernel_deriv = np.array( [[1, 0, -1],
                              [2, 0, -2],
                              [1, 0, -1]] )
    img_deriv_x = convolve( image, kernel_deriv, mode='constant', cval=0 )
    img_deriv_y = convolve( image, kernel_deriv.transpose(), mode='constant', cval=0 )

    igx = gaussian_filter( img_deriv_x**2, sigma, mode='constant', cval=0 )
    igy = gaussian_filter( img_deriv_y**2, sigma, mode='constant', cval=0 )
    igxy = gaussian_filter( img_deriv_x*img_deriv_y, sigma, mode='constant', cval=0 )

    response = (igx*igy - igxy**2) - alpha*(igx+igy)**2
    response_dilated = grey_dilation( response, size=radius )

    img_max = response == response_dilated
    img_threshed = response>thresh

    img_corner = np.logical_and(img_max,img_threshed)

    return img_corner
```

output

homework

- Write a python code that
  - Read an input image (any)
  - Find Harris corners
  - Save the result
- Write a report with procedure
- Submit your
  - Code
  - Report
  - Input image
  - Intermediate images
  - Result image