**Practical No: 8**

Page | 21

## Aim: Write the program to extract image features by implementing methods like corner and blob detectors, HoG and Haar features

### Harris Corner Detector With scikit-image : algorithm was developed to identify the internal corners of an image. The corners of an image are basically identified as the regions in which there are variations in large intensity of the gradient in all possible dimensions and directions. Corners extracted can be a part of the image features, which can be matched with features of other images, and can be used to extract accurate information. Harris Corner Detection is a method to extract the corners from the input image and to extract features from the input image.

### Program Code: A

from matplotlib import pylab as pylab

from skimage.io import imread

from skimage.color import rgb2gray

from skimage.feature import corner\_harris, corner\_subpix, corner\_peaks

from skimage.transform import warp, SimilarityTransform, AffineTransform, resize

import numpy as np

from skimage import data

from skimage.util import img\_as\_float

from skimage.exposure import rescale\_intensity

from skimage.measure import ransac

image = imread(r'C:\Users\LJP\_IT\_LAB\Pictures\pic.jpg')

image\_gray = rgb2gray(image)

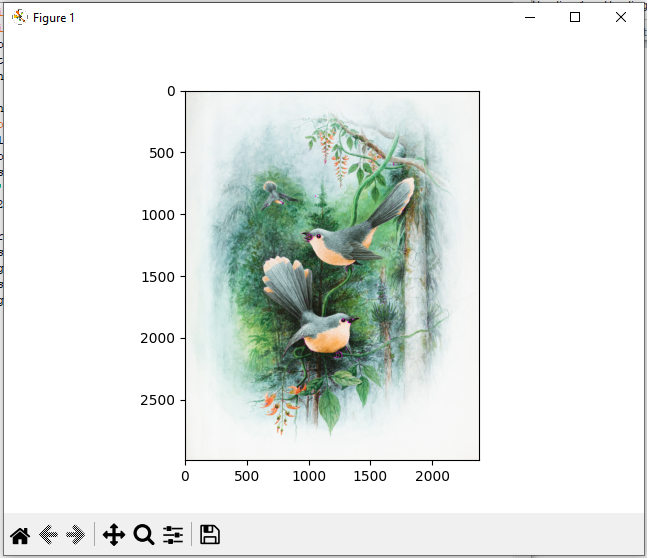
coordinates = corner\_harris(image\_gray, k =0.001)

image[coordinates>0.01\*coordinates.max()]=[255,0,0,255]

pylab.figure(figsize=(20,10))

pylab.imshow(image), pylab.axis('off'), pylab.show()

**output:**

** **

**● Blob detectors with LoG, DoG and DoH**

Page | 22

from numpy **im**por**t** sqrt

from skimage.feature import blob\_dog, blob\_log, blob\_doh

im **=** imread('../images/butterfly.png')

im\_gray **=** rgb2gray(im)

log\_blobs **=** blob\_log(im\_gray, max\_sigma**=**30, num\_sigma**=**10, threshold**=**.1)

log\_blobs[:, 2] **=** sqrt(2) **\*** log\_blobs[:, 2] *# Compute radius in the 3rd column* dog\_blobs **=** blob\_dog(im\_gray, max\_sigma**=**30, threshold**=**0.1) dog\_blobs[:, 2] **=** sqrt(2) **\*** dog\_blobs[:, 2]

doh\_blobs **=** blob\_doh(im\_gray, max\_sigma**=**30, threshold**=**0.005)

list\_blobs **=** [log\_blobs, dog\_blobs, doh\_blobs]

colors, titles **=** ['yellow','lime','red'], ['Laplacian of Gaussian', 'Difference of Gaussian', 'Determinant of Hessian'] sequence **=** zip(list\_blobs, colors, titles)

fig, axes **=** pylab**.**subplots(2, 2,figsize**=**(20, 20),sharex**=**True, sharey**=**True)

axes **=** axes**.**ravel()

axes[0]**.**imshow(im, interpolation**=**'nearest')

axes[0]**.**set\_title('original image', size**=**30), axes[0]**.**set\_axis\_off()

**for** idx, (blobs, color, title) **in** enumerate(sequence):

axes[idx**+**1]**.**imshow(im, interpolation**=**'nearest')

axes[idx**+**1]**.**set\_title('Blobs with ' **+** title, size**=**30)

**for** blob **in** blobs:

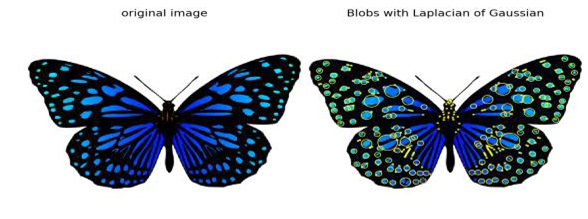
y, x, row **=** blob

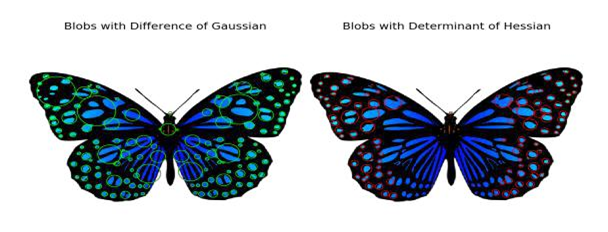
col **=** pylab**.**Circle((x, y), row, color**=**color, linewidth**=**2, fill**=**False)

axes[idx**+**1]**.**add\_patch(col), axes[idx**+**1]**.**set\_axis\_off()

pylab**.**tight\_layout(), pylab**.**show()

**Output:**





Page | 23

**Compute HOG descriptors with scikit-image**

from skimage.feature import hog

from skimage import exposure

image **=** rgb2gray(imread('../images/cameraman.jpg'))

fd,hog\_image **=** hog(image, orientations**=**8, pixels\_per\_cell**=**(16, 16),

cells\_per\_block**=**(1, 1), visualize**=**True)

print(image**.**shape, len(fd))

*# ((256L, 256L), 2048)*

fig, (axes1, axes2) **=** pylab**.**subplots(1, 2,figsize**=**(15, 10),sharex**=**True, sharey**=**True)

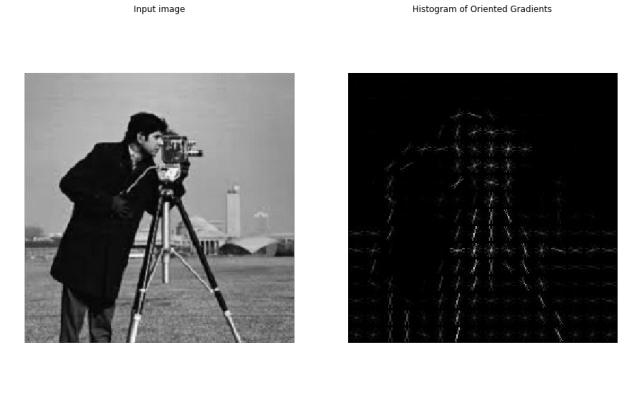
axes1**.**axis('off'), axes1**.**imshow(image, cmap**=**pylab**.**cm**.**gray), axes1**.**set\_title('Input image')

hog\_image\_rescaled **=** exposure**.**rescale\_intensity(hog\_image, in\_range**=**(0, 10))

axes2**.**axis('off'), axes2**.**imshow(hog\_image\_rescaled, cmap**=**pylab**.**cm**.**gray),

axes2**.**set\_title('Histogram of Oriented Gradients')

pylab**.**show()



**● Haar-like feature descriptor with scikit-image**

Page | 24

from skimage.feature import haar\_like\_feature\_coord

from skimage.feature import draw\_haar\_like\_feature

images **=** [np**.**zeros((2, 2)), np**.**zeros((2, 2)), np**.**zeros((3, 3)),

np**.**zeros((3, 3)), np**.**zeros((2, 2))]

feature\_types **=** ['type-2-x','type-2-y','type-3-x','type-3-y','type-4']

fig, axes **=** pylab**.**subplots(3, 2,figsize**=**(5,7))

for axes, img, feat\_t in zip(np**.**ravel(axes), images, feature\_types):

coordinates, \_ **=** haar\_like\_feature\_coord(img**.**shape[0], img**.**shape[1], feat\_t)

haar\_feature **=** draw\_haar\_like\_feature(img, 0, 0, img**.**shape[0],img**.**shape[1], coordinates, max n features**=**1, random\_state**=**0, color\_positive\_block**=**(1.0, 0.0, 0.0), color\_negative\_block**=**(0.0, 0.0, 1.0), alpha**=**0.8)



axes**.**imshow(haar\_feature), axes**.**set\_title(feat\_t), axes**.**set\_axis\_off()

*#fig.suptitle('Different Haar-like feature descriptors')*

pylab**.**axis('off'), pylab**.**tight\_layout(), pylab**.**show()

