```
import numpy as np
import matplotlib.pyplot as plt
import imageio as io
import os
import cv2 as cv



#glcm matrix return
def glcm_matrix(input_matrix, levels, distance=1, angle=0):

    glcm = np.zeros((levels, levels), dtype=int)
    rows, cols = input_matrix.shape

    # For 0-degree angle (horizontal)
    if angle == 0:
        for i in range(rows):
            for j in range(cols-distance ):
                pixel1 = input_matrix[i, j]
                pixel2 = input_matrix[i, j + distance]
                glcm[pixel1 - 1, pixel2 - 1] += 1
    # Calculate GLCM for 45-degree angle
    elif angle == 45:
      for i in range(rows):
        for j in range(cols-1):
          if (i!=0):
            pixel1 = input_matrix[i, j]
            pixel2 = input_matrix[i - distance, j + distance]
            glcm[pixel1 -1, pixel2 -1] += 1
          else:
            continue
     # Calculate GLCM for 90-degree angle
    elif angle== 90:
      for i in range(rows):
        for j in range(cols):
          if(i!=0):
            pixel1=input_matrix[i,j]
            pixel2=input_matrix[i-1,j]
            glcm[pixel1-1,pixel2-1]+=1
          else:
            continue
    elif angle== 135:
      for i in range(rows):
        for j in range(cols):
          if(i!=0 or j!=0):
            pixel1=input_matrix[i,j]
            pixel2=input_matrix[i-1,j-1]
            glcm[pixel1-1,pixel2-1]+=1
          else:
            continue









           # You can add similar blocks for other angles (45, 90, 135)

    return glcm



# Example usage:
input_matrix=np.array([[1,2,3,2,4],
                       [5,3,2,1,1],
                       [3,2,1,2,1],
                       [1,1,5,2,1],
                       [2,5,4,1,3]])

levels =np.unique(input_matrix)
```

```
levels=levels.shape[0]


glcm1 = glcm_matrix(input_matrix, levels, distance=1, angle=0)
glcm2 = glcm_matrix(input_matrix, levels, distance=1, angle=45)
glcm3 = glcm_matrix(input_matrix, levels, distance=1, angle=90)
glcm4 = glcm_matrix(input_matrix, levels, distance=1, angle=135)
print(glcm1)
print()
print(glcm2)
print()
print(glcm3)
print()
print(glcm4)
print()
```

```
[[2 2 1 0 1]
 [4 0 1 1 1]
 [0 3 0 0 0]
 [1 0 0 0 0]
 [0 1 1 1 0]]

[[3 1 0 1 0]
 [3 2 0 0 0]
 [0 0 2 0 0]
 [0 1 0 0 0]
 [0 2 0 0 1]]

[[2 4 1 1 0]
 [2 1 2 0 0]
 [1 1 0 0 1]
 [0 0 0 0 1]
 [3 0 0 0 0]]

[[2 2 3 0 1]
 [2 3 0 1 1]
 [2 1 0 0 1]
 [2 0 0 0 0]
 [1 1 0 1 0]]
```

```
def systemtric_glcm(glcm_matrix):
  transpose_glcm=np.transpose(glcm_matrix)
  final_glcm_matrix=glcm_matrix+transpose_glcm
  return final_glcm_matrix


#take systemric_matrix
systemtric_glcm=systemtric_glcm(glcm2)
systemtric_glcm
```

```
array([[6, 4, 0, 1, 0],
       [4, 4, 0, 1, 2],
       [0, 0, 4, 0, 0],
       [1, 1, 0, 0, 0],
       [0, 2, 0, 0, 2]])
```

```
#probaalistic-glcm p(i,j)
def probablistic_glcm(systemtric_glcm):
  total_gray_values=np.sum(systemtric_glcm)
  probablistic_glcm=systemtric_glcm/total_gray_values
  return probablistic_glcm


# take a probability according to total gray-values in side glcm
probablistic_glcm=probablistic_glcm(systemtric_glcm)
probablistic_glcm
```

```
array([[0.1875 , 0.125  , 0.     , 0.03125, 0.     ],
       [0.125  , 0.125  , 0.     , 0.03125, 0.0625 ],
       [0.     , 0.     , 0.125  , 0.     , 0.     ],
       [0.03125, 0.03125, 0.     , 0.     , 0.     ],
       [0.     , 0.0625 , 0.     , 0.     , 0.0625 ]])
```

```
#let start computing a 2nd order statistical feature
def energy(probablistic_glcm):
  energy=np.sum(probablistic_glcm**2)
  return energy


energy=energy(probablistic_glcm)
energy
```

```
0.11328125
```

```
#contrast
def contrast(probablistic_glcm):
  contrast=0
  for i in range(probablistic_glcm.shape[0]):
    for j in range(probablistic_glcm.shape[1]):
      contrast+=((i-j)**2)*(probablistic_glcm[i,j])
  return contrast



contrast=contrast(probablistic_glcm)
contrast
```

```
2.1875
```

```
#entropy
def entropy_func(glcm_matrix):
  entropy=0
  multiply=np.zeros((probablistic_glcm.shape[0],probablistic_glcm.shape[1]))
  for i in range(glcm_matrix.shape[0]):
    for j in range(glcm_matrix.shape[1]):
      if(glcm_matrix[i,j]!=0):
        multiply[i,j]=(probablistic_glcm[i,j]*np.log(probablistic_glcm[i,j]))
        entropy+=multiply[i,j]
      else:
        multiply[i,j]=0
        entropy+=multiply[i,j]




  return entropy,multiply


entropy,multiply=entropy_func(probablistic_glcm)
entropy,multiply
```

```
(-2.3066687254045313,
 array([[-0.31387058, -0.25993019,  0.        , -0.10830425,  0.        ],
        [-0.25993019, -0.25993019,  0.        , -0.10830425, -0.1732868 ],
        [ 0.        ,  0.        , -0.25993019,  0.        ,  0.        ],
        [-0.10830425, -0.10830425,  0.        ,  0.        ,  0.        ],
        [ 0.        , -0.1732868 ,  0.        ,  0.        , -0.1732868 ]]))
```

```
#IDM
def IDM(probablistic_glcm):
  idm=0
  multiply=np.zeros((probablistic_glcm.shape[0],probablistic_glcm.shape[1]))
  for i in range(probablistic_glcm.shape[0]):
    for j in range(probablistic_glcm.shape[1]):
      if(probablistic_glcm[i,j]!=0):
        multiply[i,j]=((1/(1+(i-j)**2)))*(probablistic_glcm[i,j])
        idm+=multiply[i,j]
      else:
        multiply[i,j]=0
        idm+=multiply[i,j]

  return idm


idm=IDM(probablistic_glcm)
idm
```

```
0.65625
```

```
a=np.arange(probablistic_glcm.shape[0] * probablistic_glcm.shape[1])
a.reshape(probablistic_glcm.shape)
```

```
array([[ 0,  1,  2,  3,  4],
       [ 5,  6,  7,  8,  9],
       [10, 11, 12, 13, 14],
       [15, 16, 17, 18, 19],
       [20, 21, 22, 23, 24]])
```

```
meanx = np.sum(np.multiply(np.arange(probablistic_glcm.shape[0] * probablistic_glcm.shape[1]).reshape(probablistic_glcm.shape), probablistic
meanx
```

```
7.6875
```

```
#correaltion
def correlation_func(probablistic_glcm):
  correlation=0

  # meanx,meany ---->sum i*px(i)
  px=np.sum(probablistic_glcm,axis=1)
  py=np.sum(probablistic_glcm,axis=0)
  meanx=np.sum(np.multiply(np.arange(px.shape[0]).reshape(px.shape),px))
  meany=np.sum(np.multiply(np.arange(py.shape[0]).reshape(py.shape),py))

  #sigmax.sigmay ---->squre root(sum(i-ux)**2(px(i)))

  sigmax=np.sqrt(np.sum(np.multiply((np.arange(px.shape[0]).reshape(px.shape)-(meanx))**2,px)))
  sigmay=np.sqrt(np.sum(np.multiply((np.arange(py.shape[0]).reshape(py.shape)-(meany))**2,py)))


  #correaltion sum((i*j)*(p(i,j)-(meanx*meany)/(sigmax*sigmay))
  for i in range(probablistic_glcm.shape[0]):
    for j in range(probablistic_glcm.shape[1]):
      correlation+=(((i*j)*probablistic_glcm[i,j])-(meanx*meany))/(sigmax*sigmay)


  return correlation



correlation=correlation_func(probablistic_glcm)
correlation
```

```
-21.946319867183185
```

```
#display a all answer for any angle
def display():
  print("Energy is ::",energy)
  print("Contrast is ::",contrast)
  print("Entropy is ::",entropy)
  print("IDM is ::",idm)
  print("Corealation is ::",correlation)


display()
```

```
Energy is :: 0.11328125
Contrast is :: 2.1875
Entropy is :: -2.3066687254045313
IDM is :: 0.65625
Corealation is :: -21.946319867183185
```