**CHAPTER 6**

**IMPLEMENTATION**

**Statistical Properties of Colour Channels**

from PIL import Image

importnumpy as np

importscipy

fromscipy import stats

import cv2

importskimage

fromskimage.measure import shannon\_entropy

importmatplotlib.pyplot as plt

import pandas as pd

classsmp\_values():

defmean\_properties(image):

#image=cv2.imread(r"C:\Users\Pavan\Desktop\Project\image\_1.jpg")

b,g,r=cv2.split(image)

rgb\_img=cv2.merge([r,g,b])

#plt.imshow(rgb\_img)

#plt.imshow(image)

red=image[:,:,2]

Mean=list()

Range=list()

Deviation=list()

Entropy=list()

Skewness=list()

Kurtosis=list()

r1=r.mean()

Mean.append(r1)

r2=skimage.measure.shannon\_entropy(r,base=10)

Entropy.append(r2)

r3=r.std()

Deviation.append(r3)

r4=np.ptp(r)

Range.append(r4)

t=scipy.stats.skew(r)

#print(b)

r5=t.mean()

Skewness.append(r5)

c=scipy.stats.kurtosis(r,axis=0,fisher=False)

#print(c)

r6=c.mean()

Kurtosis.append(r6)

g1=g.mean()

Mean.append(g1)

g2=skimage.measure.shannon\_entropy(g,base=10)

Entropy.append(g2)

g3=g.std()

Deviation.append(g3)

g4=np.ptp(g)

Range.append(g4)

e=scipy.stats.skew(g)

#print(b)

g5=e.mean()

Skewness.append(g5)

f=scipy.stats.kurtosis(g,axis=0,fisher=False)

#print(c)

g6=f.mean()

Kurtosis.append(g6)

b1=b.mean()

Mean.append(b1)

b2=skimage.measure.shannon\_entropy(b,base=10)

Entropy.append(b2)

b3=b.std()

Deviation.append(b3)

b4=np.ptp(b)

Range.append(b4)

h=scipy.stats.skew(b)

#print(b)

b5=h.mean()

Skewness.append(b5)

i=scipy.stats.kurtosis(b,axis=0,fisher=False)

#print(c)

b6=i.mean()

Kurtosis.append(b6)

#print(Mean)

#print(Entropy)

#print(Deviation)

#print(Range)

#print(Skewness)

#print(Kurtosis)

#df1= pd.DataFrame(Mean, columns = ['Mean\_R', 'Mean\_G','Mean\_B'])

df1= pd.DataFrame([Mean])

df1.columns =['Mean\_R', 'Mean\_G','Mean\_B']

df2= pd.DataFrame([Deviation])

df2.columns =['Standard\_deviation\_R', 'Standard\_deviation\_G','Standard\_deviation\_B']

df3= pd.DataFrame([Skewness])

df3.columns =['Skewness\_R', 'Skewness\_G','Skewness\_B']

df4= pd.DataFrame([Kurtosis])

df4.columns =['Kurtosis\_R', 'Kurtosis\_G','Kurtosis\_B']

df5= pd.DataFrame([Entropy])

df5.columns =['Entropy\_R', 'Entropy\_G','Entropy\_B']

df6= pd.DataFrame([Range])

df6.columns =['Range\_R', 'Range\_G','Range\_B']

frames = [df1, df2, df3, df4, df5, df6]

result = pd.concat(frames, axis=1, sort=False)

return(result)

**Gray Level Co-Occurrence Matrix (GLCM)**

importnumpy as np

import cv2

importskimage

fromskimage.feature import greycomatrix, greycoprops

import pandas as pd

classglcm\_prop():

defglclm\_properties(image):

#image=cv2.imread(r"C:\Users\Pavan\Desktop\Project\image\_1.jpg")

result=cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

#print(image)

#print(result)

#print(result.max())

angles = [0, np.pi/4, np.pi/2, 3\*np.pi/4]

entropy=[]

ASM\_val=[]

Contrast=[]

Corrleation=[]

new\_angle=0

for i in range(4):

new\_angle=angles[i]

g\_e= greycomatrix(result, [1],[new\_angle], 256, symmetric=False, normed=True)

entr = skimage.measure.shannon\_entropy(g\_e)

correlation = greycoprops(g\_e, 'correlation')

ASM = greycoprops(g\_e, 'ASM')

contrast = greycoprops(g\_e, 'contrast')

entropy.append(entr)

ASM\_val.append(ASM[0][0])

Contrast.append(contrast[0][0])

Corrleation.append(correlation[0][0])

#asm,contrast,corrleation,entropy

dummy0=[]

dummy0.append(ASM\_val[0])

dummy0.append(Contrast[0])

dummy0.append(Corrleation[0])

dummy0.append(entropy[0])

dummy45=[]

dummy45.append(ASM\_val[1])

dummy45.append(Contrast[1])

dummy45.append(Corrleation[1])

dummy45.append(entropy[1])

dummy90=[]

dummy90.append(ASM\_val[2])

dummy90.append(Contrast[2])

dummy90.append(Corrleation[2])

dummy90.append(entropy[2])

dummy135=[]

dummy135.append(ASM\_val[3])

dummy135.append(Contrast[3])

dummy135.append(Corrleation[3])

dummy135.append(entropy[3])

df1= pd.DataFrame([dummy0])

df1.columns =['ASM\_0', 'Contrast\_0','Corrleation\_0','entropy\_0']

df2= pd.DataFrame([dummy45])

df2.columns =['ASM\_45', 'Contrast\_45','Corrleation\_45','entropy\_45']

df3= pd.DataFrame([dummy90])

df3.columns =['ASM\_90', 'Contrast\_90','Corrleation\_90','entropy\_90']

df4= pd.DataFrame([dummy135])

df4.columns =['ASM\_135', 'Contrast\_135','Corrleation\_135','entropy\_135']

frames = [df1, df2, df3, df4]

result\_glcm = pd.concat(frames, axis=1, sort=False)

return(result\_glcm)

**Gray Level Run Lengths (GLRL)**

importSimpleITK as sitk

importnumpy as np

import cv2

fromskimage.color import rgb2gray

fromskimage.feature import greycomatrix, greycoprops

import pandas as pd

classglrlm\_prop():

defglrlm\_properties(image):

#image=cv2.imread(r"C:\Users\Pavan\Desktop\Project\image\_1.jpg")

grayscale = rgb2gray(image)

grayscale = np.array(grayscale)

im = sitk.GetImageFromArray(grayscale)

test\_arr = np.ones((grayscale.shape), dtype='uint8')

ma = sitk.GetImageFromArray(test\_arr)

# Store to nrrd:

sitk.WriteImage(im, 'image.nrrd')

sitk.WriteImage(ma, 'mask.nrrd', True) # enable compression to save disk space

# or extract features:

fromradiomics import featureextractor

extractor = featureextractor.RadiomicsFeatureExtractor(r'path/to/params.yml')

features = extractor.execute(im, im, label=1)

glrlm=[]

glrlm.append(float(features['original\_glrlm\_ShortRunEmphasis']))

glrlm.append(float(features['original\_glrlm\_LongRunEmphasis']))

glrlm.append(float(features['original\_glrlm\_GrayLevelNonUniformity']))

glrlm.append(float(features['original\_glrlm\_RunLengthNonUniformity']))

glrlm.append(float(features['original\_glrlm\_RunPercentage']))

glrlm.append(float(features['original\_glrlm\_LowGrayLevelRunEmphasis']))

glrlm.append(float(features['original\_glrlm\_HighGrayLevelRunEmphasis']))

glrlm.append(float(features['original\_glrlm\_ShortRunLowGrayLevelEmphasis']))

glrlm.append(float(features['original\_glrlm\_ShortRunHighGrayLevelEmphasis']))

glrlm.append(float(features['original\_glrlm\_LongRunLowGrayLevelEmphasis']))

glrlm.append(float(features['original\_glrlm\_LongRunHighGrayLevelEmphasis']))

df1= pd.DataFrame([glrlm])

df1.columns =['Short\_run\_emphasis\_0', 'Long\_run\_emphasis\_0','Gray\_level\_nonuniformity\_0' ,'Run\_length\_nonuniformity\_0','Run\_percentage\_0','Low\_gray\_level\_run\_emphasis\_0','High\_gray\_level\_run\_emphasis\_0','Short\_run\_Low\_gray\_level\_emphasis\_0','Short\_run\_High\_gray\_level\_emphasis\_0','Long\_run\_Low\_gray\_level\_emphasis\_0','Long\_run\_High\_gray\_level\_emphasis\_0']

df2= pd.DataFrame([glrlm])

df2.columns =['Short\_run\_emphasis\_45', 'Long\_run\_emphasis\_45','Gray\_level\_nonuniformity\_45' ,'Run\_length\_nonuniformity\_45','Run\_percentage\_45','Low\_gray\_level\_run\_emphasis\_45','High\_gray\_level\_run\_emphasis\_45','Short\_run\_Low\_gray\_level\_emphasis\_45','Short\_run\_High\_gray\_level\_emphasis\_45','Long\_run\_Low\_gray\_level\_emphasis\_45','Long\_run\_High\_gray\_level\_emphasis\_45']

df3= pd.DataFrame([glrlm])

df3.columns =['Short\_run\_emphasis\_90', 'Long\_run\_emphasis\_90','Gray\_level\_nonuniformity\_90' ,'Run\_length\_nonuniformity\_90','Run\_percentage\_90','Low\_gray\_level\_run\_emphasis\_90','High\_gray\_level\_run\_emphasis\_90','Short\_run\_Low\_gray\_level\_emphasis\_90','Short\_run\_High\_gray\_level\_emphasis\_90','Long\_run\_Low\_gray\_level\_emphasis\_90','Long\_run\_High\_gray\_level\_emphasis\_90']

df4= pd.DataFrame([glrlm])

df4.columns =['Short\_run\_emphasis\_135', 'Long\_run\_emphasis\_135','Gray\_level\_nonuniformity\_135' ,'Run\_length\_nonuniformity\_135','Run\_percentage\_135','Low\_gray\_level\_run\_emphasis\_135','High\_gray\_level\_run\_emphasis\_135','Short\_run\_Low\_gray\_level\_emphasis\_135','Short\_run\_High\_gray\_level\_emphasis\_135','Long\_run\_Low\_gray\_level\_emphasis\_135','Long\_run\_High\_gray\_level\_emphasis\_135']

frames = [df1, df2, df3, df4]

result\_glrlm = pd.concat(frames, axis=1, sort=False)

return(result\_glrlm)

**Support Vector Machine (SVM)**

import pickle

header\_list =['Mean\_R', 'Mean\_G','Mean\_B','Standard\_deviation\_R', 'Standard\_deviation\_G','Standard\_deviation\_B','Skewness\_R', 'Skewness\_G','Skewness\_B','Kurtosis\_R', 'Kurtosis\_G','Kurtosis\_B','Entropy\_R', 'Entropy\_G','Entropy\_B','Range\_R', 'Range\_G','Range\_B','ASM\_0', 'Contrast\_0','Corrleation\_0','entropy\_0','ASM\_45', 'Contrast\_45','Corrleation\_45','entropy\_45','ASM\_90', 'Contrast\_90','Corrleation\_90','entropy\_90','ASM\_135', 'Contrast\_135','Corrleation\_135','entropy\_135','Short\_run\_emphasis\_0', 'Long\_run\_emphasis\_0','Gray\_level\_nonuniformity\_0' ,'Run\_length\_nonuniformity\_0','Run\_percentage\_0','Low\_gray\_level\_run\_emphasis\_0','High\_gray\_level\_run\_emphasis\_0','Short\_run\_Low\_gray\_level\_emphasis\_0','Short\_run\_High\_gray\_level\_emphasis\_0','Long\_run\_Low\_gray\_level\_emphasis\_0','Long\_run\_High\_gray\_level\_emphasis\_0','Short\_run\_emphasis\_45', 'Long\_run\_emphasis\_45','Gray\_level\_nonuniformity\_45' ,'Run\_length\_nonuniformity\_45','Run\_percentage\_45','Low\_gray\_level\_run\_emphasis\_45','High\_gray\_level\_run\_emphasis\_45','Short\_run\_Low\_gray\_level\_emphasis\_45','Short\_run\_High\_gray\_level\_emphasis\_45','Long\_run\_Low\_gray\_level\_emphasis\_45','Long\_run\_High\_gray\_level\_emphasis\_45','Short\_run\_emphasis\_90', 'Long\_run\_emphasis\_90','Gray\_level\_nonuniformity\_90' ,'Run\_length\_nonuniformity\_90','Run\_percentage\_90','Low\_gray\_level\_run\_emphasis\_90','High\_gray\_level\_run\_emphasis\_90','Short\_run\_Low\_gray\_level\_emphasis\_90','Short\_run\_High\_gray\_level\_emphasis\_90','Long\_run\_Low\_gray\_level\_emphasis\_90','Long\_run\_High\_gray\_level\_emphasis\_90','Short\_run\_emphasis\_135', 'Long\_run\_emphasis\_135','Gray\_level\_nonuniformity\_135' ,'Run\_length\_nonuniformity\_135','Run\_percentage\_135','Low\_gray\_level\_run\_emphasis\_135','High\_gray\_level\_run\_emphasis\_135','Short\_run\_Low\_gray\_level\_emphasis\_135','Short\_run\_High\_gray\_level\_emphasis\_135','Long\_run\_Low\_gray\_level\_emphasis\_135','Long\_run\_High\_gray\_level\_emphasis\_135','Result']

import pandas as pd

fromsklearn.model\_selection import train\_test\_split

df = pd.read\_csv("dataset.csv", names=header\_list)

X\_train\_1=df.iloc[:,:-1]

Y\_train\_1=df.iloc[:,-1]

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_train\_1,Y\_train\_1, test\_size=0.2)

fromsklearn import svm

classifier = svm.SVC(kernel='linear')

classifier.fit(X\_train, y\_train)

#predicting Y values for test data

Y\_pred = classifier.predict(X\_test)

#confusion matrix for tested datatsets

fromsklearn.metrics import confusion\_matrix

cm = confusion\_matrix(y\_test, Y\_pred)

#Store the trainined SVM Model for future use

pickle.dump(classifier,open(r'C:\Users\Pavan\Desktop\Project\corrosion\_detection\predictor\model\model.pkl', 'wb'))