Importarea modulelor

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

from skimage.io import imshow, imread

from sklearn import datasets, ensemble

import matplotlib.pyplot as plt
import numpy as np
from sklearn import svm, metrics, datasets

from numpy import pi

from skimage.feature import hog
from skimage import data, exposure
from skimage.color import rgb2gray

import os
```

Partea de parsare

In [2]:
```
dir_list = os.listdir('Cohn Kanade_annotations')
```

Datele initiale

In [3]:
```
negative = ['F', 'G', 'A', 'D']
positive = ['H', 'S']
etichete= []
date = []
```

Parsarea etichetelor

In [4]:
```
emotii = []
for pers in dir_list:
    dir_list_pers = os.listdir('Cohn Kanade_annotations'+ '/' +pers)
    # print('Person {}: {}'.format(pers, dir_list_pers))
    emotii_pers = []
    for emotion in dir_list_pers:

        if emotion[-1] in negative:
            emotii_pers.append('N')
        else:
            emotii_pers.append('P')
    emotii.append(emotii_pers)
```

Parsarea imaginilor

In [5]:
```
lista_imagini = []
for index1, pers in enumerate(dir_list):
    dir_list_pers = os.listdir('Cohn_Kanade_images'+ '/' +pers)
    for index2, emotion in enumerate(dir_list_pers):
        dir_list_images = os.listdir('Cohn_Kanade_images'+ '/' +pers + '/' +emotion)
        for image in dir_list_images[5:]:
            if image == 'Thumbs.db':
                continue
            curr_image = imread('Cohn_Kanade_images'+ '/' +pers + '/' +emotion + '/' +
            lista_imagini.append(curr_image)
            etichete.append(emotii[index1][index2])
```

Conversia imaginilor in grayscale

In [6]:
```
lista_imagini_grayscale = []
for imagine in lista_imagini:
    if len(imagine.shape) == 2:
        lista_imagini_grayscale.append(imagine)
    else:
        gray = rgb2gray(imagine)
        lista_imagini_grayscale.append(gray)
```

Taierea imaginilor

OpenCV automat

In [7]:
```
import cv2
cropped_list = []
index = 0
for imagine in lista_imagini_grayscale:

    image = np.array(imagine, dtype='uint8')

    # Load the cascade
    face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalfa

    # Detect faces
    faces = face_cascade.detectMultiScale(image, 1.1, 4)

    # we assume an error of 0.03% that  the detected facec is not a real face
    if len(faces) != 1:
        etichete.pop(index)
        continue
    index += 1
    # Draw rectangle around the faces and crop the faces
    for (x, y, w, h) in faces:
        cv2.rectangle(image, (x, y), (x+w, y+h), (0, 0, 255), 2)
        faces = image[y:y + h, x:x + w]
        resized = cv2.resize(faces, dsize=(350, 350), interpolation=cv2.INTER_CUBIC)
        cropped_list.append(resized)
```

Aplicarea HOG

In [8]:
```
imagini_hog = []
for imagine in cropped_list:
    fd, hog_image = hog(imagine, orientations=9, pixels_per_cell=(10, 10),
                        cells_per_block=(1, 1), visualize=True)
    date.append(fd)
    imagini_hog.append(hog_image)
```

Impartirea datelor

In [9]:
```
from sklearn.model_selection import train_test_split

# Split dataset into training set and test set
date_train, date_test, etichete_train, etichete_test = train_test_split(date, etichete,
print(len(date_train), len(date_test))
print(len(etichete_train), len(etichete_test))
```

```
4316 1850
4316 1850
```

Antrenarea si testarea algoritmului

In [19]:
```
clf = svm.SVC(kernel='linear',C=0.1, gamma=10)
clf.fit(date_train,etichete_train)
predictii=clf.predict(date_test)

acuratete = metrics.accuracy_score(y_true = etichete_test, y_pred = predictii)
print('Acuratete test = ',acuratete)
```

```
Acuratete test =  0.9794594594594594
```