**USTHB** **Année 2019/2020**
**Faculté d'Electronique et Informatique** **Sécurisation des**
**Département Informatique** **données et biométrie**
**Master 2 BioInformatique**

**Projet de TP**

---

**NB : Le projet doit être remis avant les vacances d'hiver, au plus tard, le 22 décembre. Il doit contenir le code source, un exécutable et un rapport de projet. Le code doit être lisible et bien commenté**
**Le projet peut se faire en binôme ou monôme. La liste des binômes doit être constituée dès la 1ère semaine de réception de cet énoncé.**

## Enoncé

On s'intéresse à la réalisation d'une application graphique permettant de simuler un système de contrôle biométrique permettant d'authentifier une personne à l'aide d'une caméra.

L'application doit être constituée de deux parties :

1. Une partie permettant l'enregistrement d'une photo de la personne dans un répertoire spécifique, constituant la base de données du système biométrique.

2. Une partie permettant d'acquérir une photo dans un but d'identification, puis d'identifier si la photo capturée correspond à une des photos de votre répertoire (BD). Cette partie fait appel à un traitement mathématique qui va comparer la photo capturée avec toutes celles qui existent dans la base de données. Si une des photos correspond, l'application devra donner accès à une page d'informations, sinon un message d'erreur est affiché.

## Langage et environnement

Il est recommandé d'utiliser **Python**, car il y a plein de code sur le net que vous pouvez réutiliser intelligemment.

## Capture d'images

La capture d'image peut se faire à l'aide de la librairie OpenCV de Python, ainsi que le package Matplotlib qui permet de visualiser les images (consulter https://github.com/kevinam99/capturing-images-from-webcam-using-opencv-python ).

## Comparaison d'images

Il est possible de comparer 2 images de deux manières (consulter https://www.pyimagesearch.com/2014/09/15/python-compare-two-images/ ). Vous pouvez utiliser la méthode qui vous convient :

1. **Algorithme de l'Erreur quadratique moyenne (Mean Squared Error MSE)**
   L'équation de l'erreur quadratique est donnée comme suit :

$$MSE = \frac{1}{m\,n}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}[I(i,j) - K(i,j)]^2$$

Cette équation peut être implémentée à l'aide de la fonction Python suivante :

```
def mse(imageA, imageB):
    # the 'Mean Squared Error' between the two images is the sum of the squared
    # difference between the two images;
    # NOTE: the two images must have the same dimension
    # Convert the images from unsigned 8-bit integers to floating point
    # Then, take the difference between the images by subtracting the pixel intensities
    # Finally, square these differences and sum them up
    err = np.sum((imageA.astype("float") - imageB.astype("float")) ** 2)
    # Divide sum of squares by the total number of pixels in the image
    err /= float(imageA.shape[0] * imageA.shape[1])
    # return the MSE, the lower the error, the more "similar" the two images are
    return err
```

Lorsque l'erreur retournée par l'algorithme est très petite, l'application peut accepter l'identification. Le seuil d'acceptation doit être défini dans votre application manuellement. Il doit être possible de changer ce seuil pour permettre de varier les tests.
Il faut savoir qu'une valeur nulle de l'erreur MSE indique une parfaite similarité des deux images comparées.
*L'inconvénient de cette méthode réside dans le fait qu'une erreur très grande, indiquant des distances larges entre les images, n'implique pas nécessairement que les images sont très différentes.*

## 2. Algorithme de l'indice de similarité structurelle (Structural SImilarity Measure SSIM)

Développée par Wang et al. l'équation de l'index de similarité est donnée par :

$$\mathrm{SSIM}(x,y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}$$

Elle permet de modéliser les changements perçus dans l'information structurelle des deux images. De plus, elle peut être utilisée pour comparer deux fenêtres (petites parties) des images et non les images complètes, ce qui est plus intéressant en biométrie pour extraire les parties les plus significatives. Cette approche est plus robuste.

Les paramètres de l'équation incluent la localisation (x,y) de la fenêtre NxN dans chaque image, la moyenne des intensités des pixels dans le directions x et y, et la variance entre ces intensités ainsi que leur covariance.

L'index de similarité obtenu sera entre -1 et 1, où 1 désignera une parfaite similarité. L'implémentation de cet algorithme est donnée sur https://scikit-image.org/.