

## TP boosting

### ≡ Première partie: comprendre et modifier *facetedetect*

La première partie de ce TP est basée sur l'exemple *facetedetect* d'OpenCV (existe en C++ et python).

- Tester les différentes options (sur ligne de commande)
- A quoi servent les fichiers xml de cascade ou "nested cascades" ?
- Comment accélérer le traitement de chaque frame (à l'aide des options) ?
- Quelle idée pourrait-on avoir pour accélérer encore plus le traitement de chaque frame ?

Lire le code (C++ ou python).

- Modifier le code pour pouvoir afficher un nez rouge, un chapeau sur la tête ou autre accessoire de votre choix.
- Faites une capture de votre meilleure réalisation.
- Réalisez votre premier rendu de TP

### ≡ Deuxième partie: un apprentissage de *boosting*



Pour cela, vous allez utiliser la librairie python scikit-learn

```
pip3 install sklearn
```

ainsi que les données MNIST par exemple:

```
from keras.datasets import mnist
```

```
(x_train, y_train), (x_test, y_test) = mnist.load_data()
```

dont vous extrairez 2 classes au choix.

Le boosting demande que l'on choisisse des classifieurs faibles. Vous choisirez les filtres de Haar (feature.haar\_like\_feature\_coord).

Quels sont les différents types de filtre selon Haar ?

Dans le cas de MNIST, combien de filtres obtenez-vous ?

Comment réduire le nombre de filtres ?

Affichez la matrice de confusion de vos résultats de classification.



[ACTIVITÉ PRÉCÉDENTE](#)  
[slides du cours boosting](#)

[ACTIVITÉ SUIVANTE](#)  
[OpenCV4 : tous les exemples python](#)

