# TP3 : Filtre Moyenneur et gaussien par convolution
## Détecteurs de contours, gradient et laplacien
## Bruit gaussien- Bruit poivre et sel

### Notebook 1 : (Suite TP2) Filtre Moyenneur et gaussien par convolution

**Instructions :**

- Charger l'image depuis **: skimage**.data.camera( )
- Utilisez l'opération de convolution 2D pour appliquer un filtre moyenneur de taille 15x15.
- Générez un masque de convolution pour un filtre gaussien de taille 31x31 et $\sigma$ = 7.
- Essayez différentes valeurs pour bien observer leur influence.

---

### Rappel : filtre moyenneur

Plus d'informations sur l'opérateur de convolution scipy.signal.convolve2d. Notamment, pour conserver la taille de l'image d'origine, il faut utiliser l'option **mode='same'**.

**Rappel** : pour générer une matrice de valeurs constantes, utilisez :
h = 1/9*np.array([[1,1,1],[1,1,1],[1,1,1]])

ou plus généralement : taille = 3
h = np.ones((taille, taille)) / taille**2

### Rappel : filtre gaussien

- **Formulation discrète** 3 × 3

$$h = \frac{1}{16}\begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix}$$

- **Formulation générale de taille** $2k + 1 \times 2k + 1$ **(***k* entier)

Utilisez la fonction np.meshgrid pour définir une fonction de 2 variables. La formule générale pour une matrice *h* définie par ses coefficients *hij* est

$$h_{ij} = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{(i-(k+1))^2 + (j-(k+1))^2}{2\sigma^2}\right)$$

## Notebook 2 : Détecteurs de contours, gradient et laplacien

— Créez des masque de convolution correspondant au filtre gradient puis au filtre laplacien.

— Utilisez l'opération de convolution 2D pour appliquer un filtre gradient (dans chaque direction), puis calculez la norme.

— Utilisez l'opération de convolution 2D pour appliquer un filtre laplacien.

## Notebook 3 : Bruit gaussien- Bruit poivre et sel

—Importez et affichez une image (parapente.png)

— Ajoutez du bruit poivre & sel avec la fonction skimage.util.random_noise, et observez l'influence des paramètres de la fonction.

— Affichez l'histogramme sur une zone homogène de l'image, et retrouvez la proportion de pixels défaillants dans l'image.

NB : pour compter le nombre de pixels à 0 dans une image x, vous pouvez utiliser l'instruction N =np.sum(x==0)