Activite - Classification Donnees Visuelles

May 9, 2018

1 Activité - Classification de données visuelles

1.1 0. Import des modules Python

In [15]:
```
# -*- coding: utf-8 -*-
from PIL import Image
from PIL import ImageFilter
from PIL.ImageFilter import (GaussianBlur,MedianFilter)

import numpy as np
import matplotlib.pyplot as plt
import os
import errno
from os import listdir
from os.path import isfile, join
```

1.2 1. Création de la fonction de filtre

1.2.1 Définition des paramètres

In [14]:
```
# Définitions des constantes de l'apllication
BAD_SYNTAX_FILTER = 'Le nom du filtre que vous avez saisi est incorrect... Réessayez S
SAVING_DIRECTORY = 'C:\\Users\\monne\\Desktop\\Pictures\\'
SOURCE_DIRECTORY = 'E:\Data\RawData\Cours_Vision_par_ordinateur-Images'
```

In [ ]:
```
# Définition des paramètres des filtres qui seront appliqués aux images
param_filtre_moy1 = {
                     'nom_filtre':'moyenneur',
                     'nom_combinaison_parametres':'noyau_dim_3_const',
                     'noyau':(np.ones(9)),
                     'taille':3,
                     'ponderation':3*3,
                     'bordure':0
                     }

param_filtre_moy2 = {
                     'nom_filtre':'moyenneur',
                     'nom_combinaison_parametres':'noyau_dim_5_const',
```

```
                    'noyau':(np.ones(25)),
                    'taille':5,
                    'ponderation':5*5,
                    'bordure':0
                    }

param_filtre_moy3 = {
                    'nom_filtre':'moyenneur',
                    'nom_combinaison_parametres':'noyau_dim_3_prog',
                    'noyau':(np.linspace(-4,4,9)),
                    'taille':3,
                    'ponderation':1,#np.size(np.linspace(-4,4,9)),
                    'bordure':0
                    }

param_filtre_moy4 = {
                    'nom_filtre':'moyenneur',
                    'nom_combinaison_parametres':'noyau_dim_5_prog',
                    'noyau':(np.linspace(-12,12,25)),
                    'taille':5,
                    'ponderation':1,#np.size(np.linspace(-12,12,25)),
                    'bordure':0
                    }

param_filtre_moy5 = {
                    'nom_filtre':'moyenneur',
                    'nom_combinaison_parametres':'noyau_dim_3_tuple',
                    'noyau':(-2,-1,0,-1,1,1,0,1,2),
                    'taille':3,
                    'ponderation':sum((-2,-1,0,-1,1,1,0,1,2)),#np.size(np.linspace(-12
                    'bordure':0
                    }

param_filtre_moy6 = {
                    'nom_filtre':'moyenneur',
                    'nom_combinaison_parametres':'noyau_dim_3_random',
                    'noyau':(np.random.randint(-5,5,9)),
                    'taille':3,
                    'ponderation':1,
                    'bordure':0
                    }

param_filtre_moy7 = {
                    'nom_filtre':'gaussien',
                    'nom_combinaison_parametres':'radius_blur_1',
                    'radius':1
                    }
```

```
param_filtre_moy8 = {
                    'nom_filtre':'gaussien',
                    'nom_combinaison_parametres':'radius_blur_2',
                    'radius':2
                    }

param_filtre_moy9 = {
                    'nom_filtre':'gaussien',
                    'nom_combinaison_parametres':'radius_blur_3',
                    'radius':3
                    }

param_filtre_moy10 = {
                    'nom_filtre':'gaussien',
                    'nom_combinaison_parametres':'radius_blur_4',
                    'radius':4
                    }

param_filtre_moy11 = {
                    'nom_filtre':'gaussien',
                    'nom_combinaison_parametres':'radius_blur_5',
                    'radius':5
                    }

param_filtre_moy12 = {
                    'nom_filtre':'gaussien',
                    'nom_combinaison_parametres':'radius_blur_6',
                    'radius':6
                    }

param_filtre_moy13 = {
                    'nom_filtre':'median',
                    'nom_combinaison_parametres':'size_9',
                    'size':9
                    }

param_filtre_moy14 = {
                    'nom_filtre':'median',
                    'nom_combinaison_parametres':'size_25',
                    'size':25
                    }

param_filtre_moy15 = {
                    'nom_filtre':'median',
                    'nom_combinaison_parametres':'size_49',
                    'size':49
                    }
```

```
param_filtre_moy16 = {
                     'nom_filtre':'median',
                     'nom_combinaison_parametres':'size_81',
                     'size':81
                     }

param_filtre_moy17 = {
                     'nom_filtre':'median',
                     'nom_combinaison_parametres':'size_121',
                     'size':121
                     }

param_filtre_moy18 = {
                     'nom_filtre':'median',
                     'nom_combinaison_parametres':'size_169',
                     'size':169
                     }

liste_parametres = [param_filtre_moy1,
                    param_filtre_moy2,
                    param_filtre_moy3,
                    param_filtre_moy4,
                    param_filtre_moy5,
                    param_filtre_moy6,
                    param_filtre_moy7,
                    param_filtre_moy8,
                    param_filtre_moy9,
                    param_filtre_moy10,
                    param_filtre_moy11,
                    param_filtre_moy12,
                    param_filtre_moy13,
                    param_filtre_moy14,
                    param_filtre_moy15,
                    param_filtre_moy16,
                    param_filtre_moy17,
                    param_filtre_moy18
                    ]
```

1.2.2 Définition de la fonction d'application du filtre sur une image

```
In [ ]: # Liste des fichiers à tester
        def Liste_Fichiers_Images_Test(chemin):
            # Renvoie le nom des fichiers image à tester
            liste_fichiers = [f for f in listdir(chemin) if isfile(join(chemin, f))]
            return liste_fichiers

        # Applique le filtre passé en paramètre à une image donné également en paramètre
        def Apply_Filter_Picture(nameFile,directory,parametres_filtre):
```

```
File='\\'.join([SOURCE_DIRECTORY,nameFile])
img = Image.open(File)

nomFichier = (SAVING_DIRECTORY
                  +directory+ '\\'
                  +parametres_filtre['nom_filtre']
                  +'_'
                  +parametres_filtre['nom_combinaison_parametres']
                  +'_'
                  + nameFile)

# Filtre de type Moyenneur
if parametres_filtre['nom_filtre']=='moyenneur':
    # Noyau de convolution du filtre moyenneur
    noyau = parametres_filtre['noyau']
    # Taille du masque de convolution
    taille = parametres_filtre['taille']
    # Paramètre par défaut pour les pixels de bords d'image
    bordure = parametres_filtre['bordure']
    # facteur de ponderation du noyau de convolution
    ponderation = parametres_filtre['ponderation']

    filtre = ImageFilter.Kernel(
                                size=(taille, taille),
                                kernel=noyau,
                                scale=ponderation,
                                offset=bordure)

# Filtre de type Gaussien
elif parametres_filtre['nom_filtre']=='gaussien':
    # Rayon du noyau gaussien
    radius=parametres_filtre['radius']
    filtre = ImageFilter.GaussianBlur(radius)

# Filtre de type médian
elif parametres_filtre['nom_filtre']=='median':
    # Taille du voisinage pour calcul de la médiane
    taille=parametres_filtre['size']
    filtre = ImageFilter.MedianFilter(size=taille)

# Non de filtre non reconnu
else:
    print(BAD_SYNTAX_FILTER)

img.filter(filtre).save(nomFichier)
```

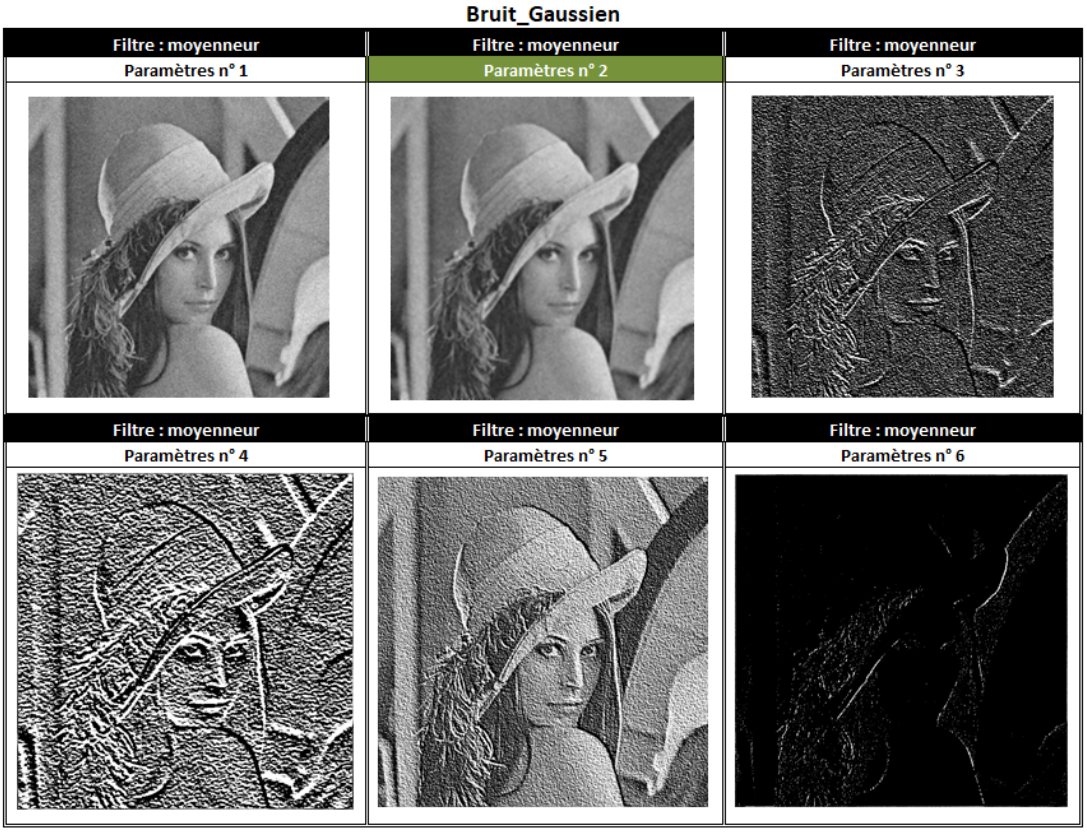
1.3 2. Comparaison des paramètres

```
In [ ]: # Application des filtres avec combinaison des paramètres sur les images bruitées
        for fichier in Liste_Fichiers_Images_Test(SOURCE_DIRECTORY):
                # Nom du répertoire associé à l'image testé
                nomRepertoire=fichier[0].upper()+fichier[1:fichier.find('.')]
                # Création du répertoire si'il n'existe pas encore
                try:
                    os.chdir(SAVING_DIRECTORY)
                    os.makedirs(nomRepertoire)
                except OSError as exception:
                    if exception.errno != errno.EEXIST:
                        raise
                # Création des nouvelles images après application des filtres sur le cliché or
                for param in liste_parametres:
                    Apply_Filter_Picture(fichier,nomRepertoire,param)
```

1.3.1 Filtre de type Moyenneur

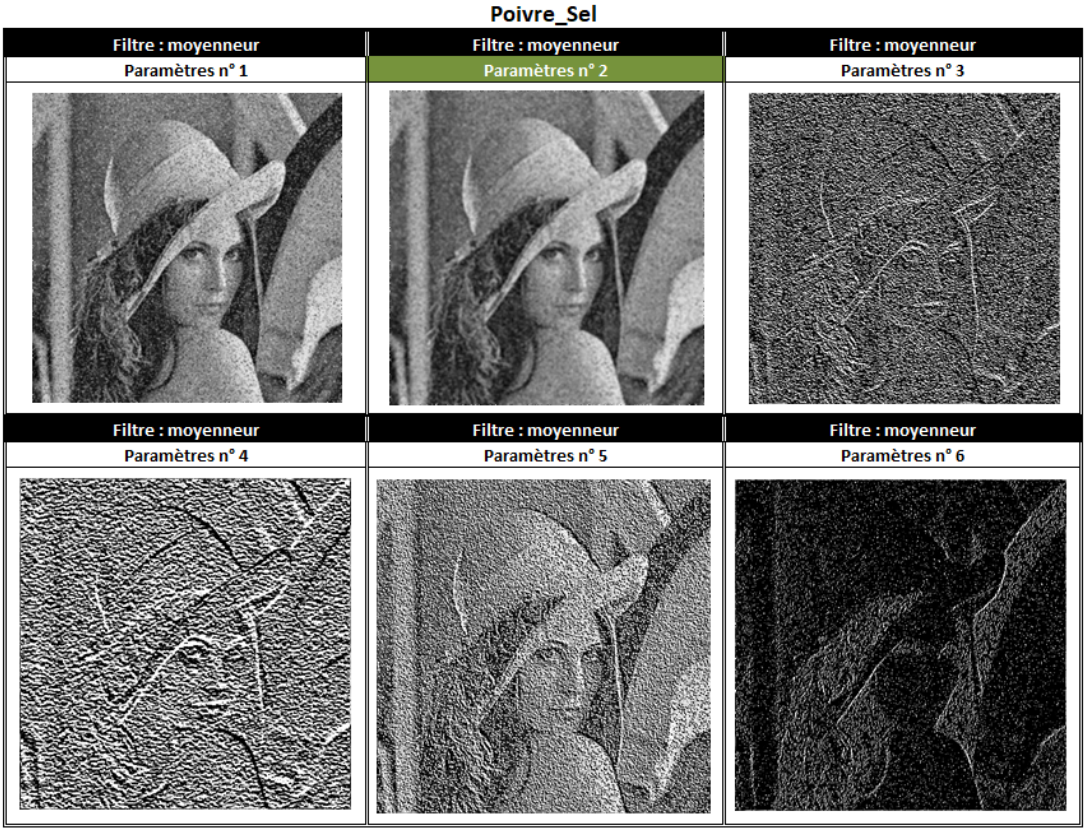
```
In [1]: from IPython.display import Image
        from IPython.core.display import HTML

        Image(filename = "Comparatifs\Moyenneur_Bruit_Gaussien.png")
```

Out[1]:



In [2]: Image(filename = "Comparatifs\Moyenneur_Poivre_Sel.png")

Out[2]:



In [3]: Image(filename = "Comparatifs\Moyenneur_Speckle.png")

Out[3]:

Speckle

Filtre : moyenneur Paramètres n° 1	Filtre : moyenneur Paramètres n° 2	Filtre : moyenneur Paramètres n° 3
		
Filtre : moyenneur **Paramètres n° 4**	**Filtre : moyenneur** **Paramètres n° 5**	**Filtre : moyenneur** **Paramètres n° 6**
		

1.3.2 Filtre de type Gaussien

```
In [4]: Image(filename = "Comparatifs\Gaussien_Bruit_Gaussien.png")
```

Out[4]:

Bruit_Gaussien

Filtre : gaussien radius = 1	Filtre : gaussien radius = 2	Filtre : gaussien radius = 3
		
Filtre : gaussien **radius = 4**	**Filtre : gaussien** **radius = 5**	**Filtre : gaussien** **radius = 6**
		

```
In [5]: Image(filename = "Comparatifs\Gaussien_Poivre_Sel.png")
```

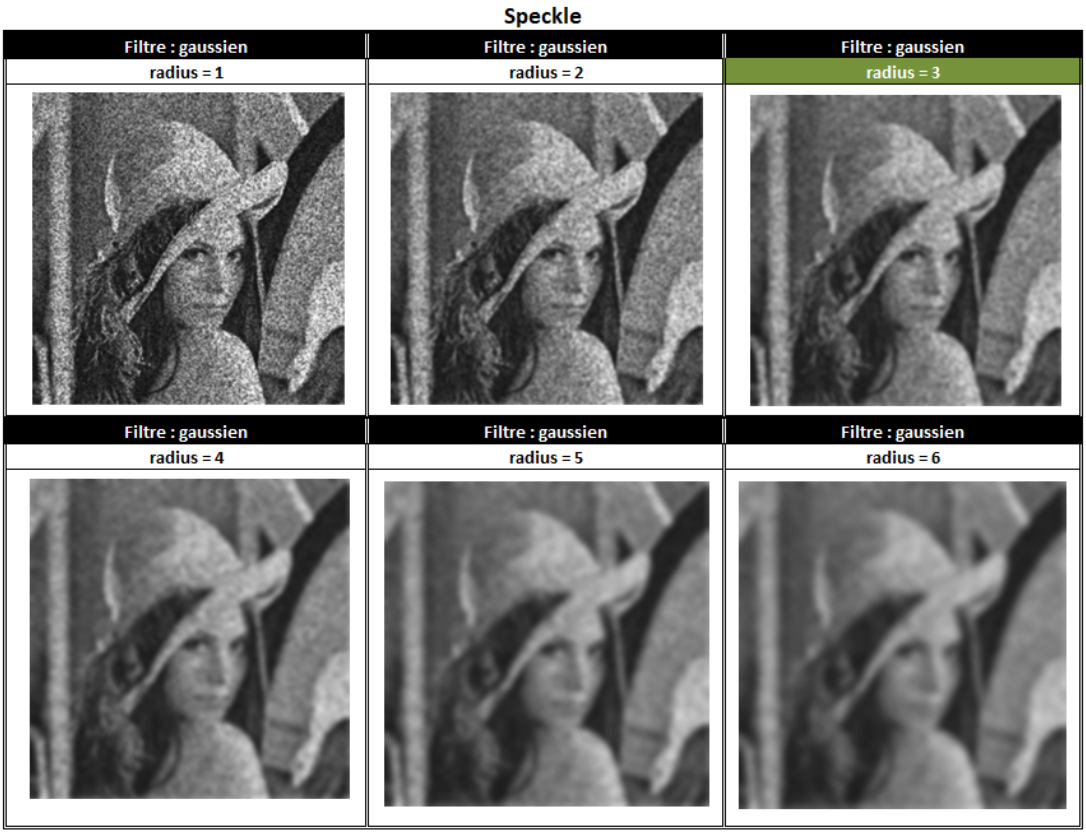
Out[5]:

Poivre_Sel

Filtre : gaussien radius = 1	Filtre : gaussien radius = 2	Filtre : gaussien radius = 3
		
Filtre : gaussien radius = 4	**Filtre : gaussien** radius = 5	**Filtre : gaussien** radius = 6
		

```
In [6]: Image(filename = "Comparatifs\Gaussien_Speckle.png")
```

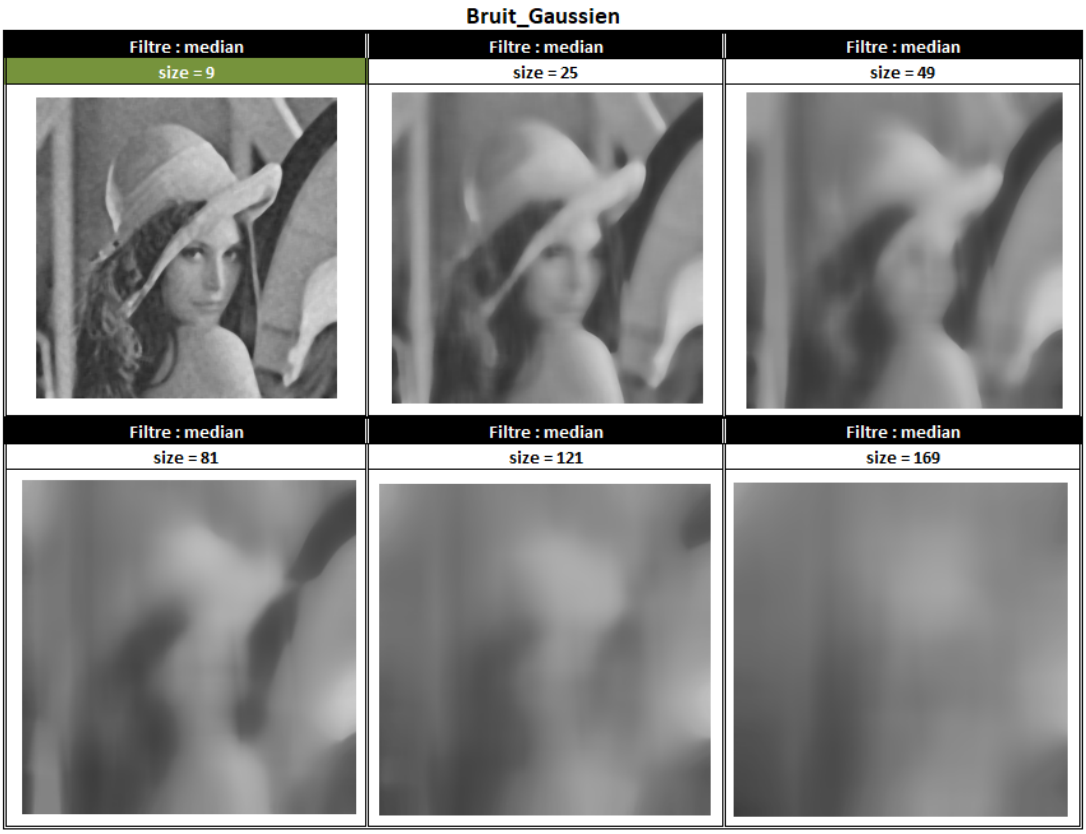
Out[6]:



1.3.3 Filtre de type médian

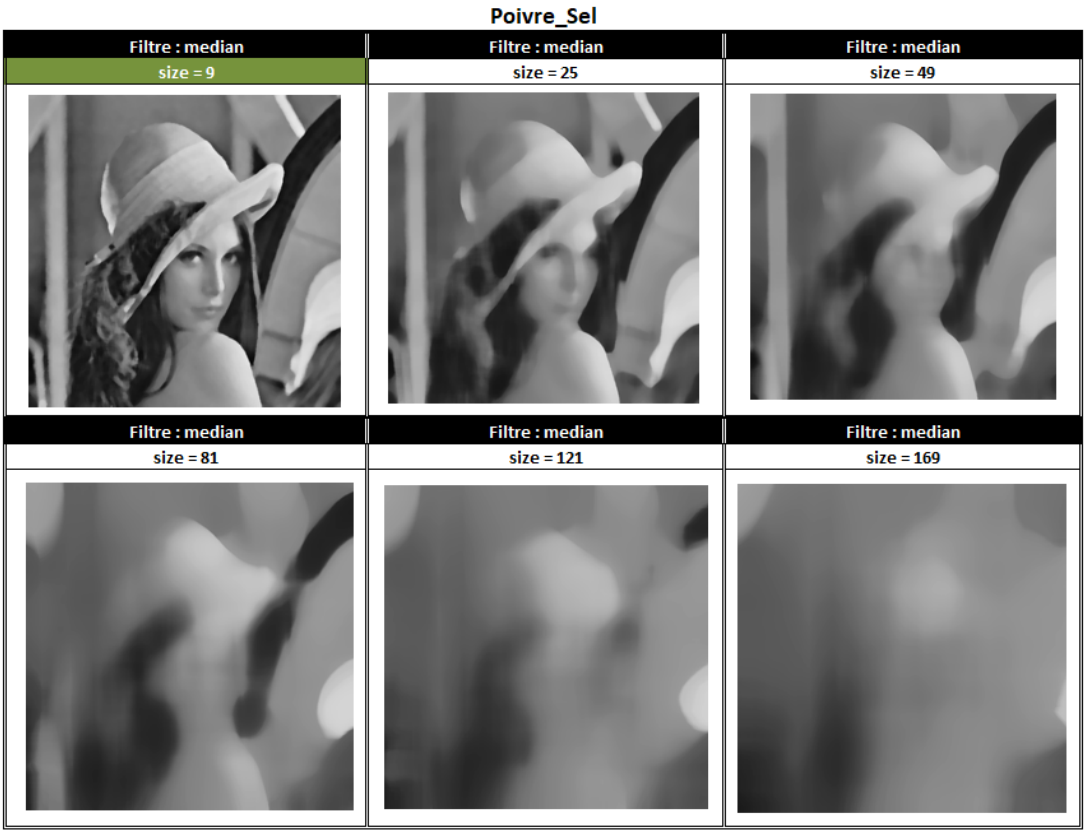
```
In [7]: Image(filename = "Comparatifs\Median_Bruit_Gaussien.png")
```

Out[7]:




```
In [8]: Image(filename = "Comparatifs\Median_Poivre_Sel.png")
```

Out[8]:




```
In [9]: Image(filename = "Comparatifs\Median_Speckle.png")
```

Out[9]:

Speckle

Filtre : median size = 9	Filtre : median size = 25	Filtre : median size = 49
		
Filtre : median **size = 81**	**Filtre : median** **size = 121**	**Filtre : median** **size = 169**
		

1.3.4 Comparatif des combinaisons de paramètres

```
In [10]: Image(filename = "Comparatifs\Choix_Parametres_Optimaux.png")
```

Out[10]:

Choix des paramètres optimaux

Filtre	Paramètres	Images bruitées			Choix	Choix
		Bruit Gaussien	Poivre & Sel	Speckle		
Moyenneur	*Taille 3 x 3 - Noyau 1...1*				*Moyenneur* *Taille 3 x 3* *Noyau 1..1* *Pondération 1/9*	*L'augmentation de la taille du noyau améliore la correction. En revanche, des valeurs de noyau dispersées brouille complètement l'image...*
	Taille 5 x 5 - Noyau 1...1	X	X	X		
	Taille 3 x 3 - Noyau -4..4					
	Taille 5 x 5 - Noyau -12..12					
	Taille 3 x 3 - Noyau tuple					
	Taille 3 x 3 - Noyau aléatoire					
Gaussien	*Rayon = 1*				*Gaussien* *Rayon=2*	*L'augmentation du rayon du noyau gaussien floute l'image jusqu'à la défigurer complètement...*
	Rayon = 2		X	X		
	Rayon = 3	X				
	Rayon = 4					
	Rayon = 5					
	Rayon = 6					
Médian	*Taille voisinage = 9*	X	X	X	*Médian* *Taille voisinage = 9*	*L'augmentation de la taille du voisinage pour le calcul de la médiane dilue complètement les zones de couleur...*
	Taille voisinage = 25					
	Taille voisinage = 49					
	Taille voisinage = 81					
	Taille voisinage = 121					
	Taille voisinage = 169					

1.4 3. Choix du filtre optimal

```
In [11]: Image(filename = "Comparatifs\Choix_Filtre_Final.png")
```

Out[11]:

1.4.1 Conclusion :

Au final, il semblerait que chaque type de filtre soit adapté à un type de bruitage spécifique :

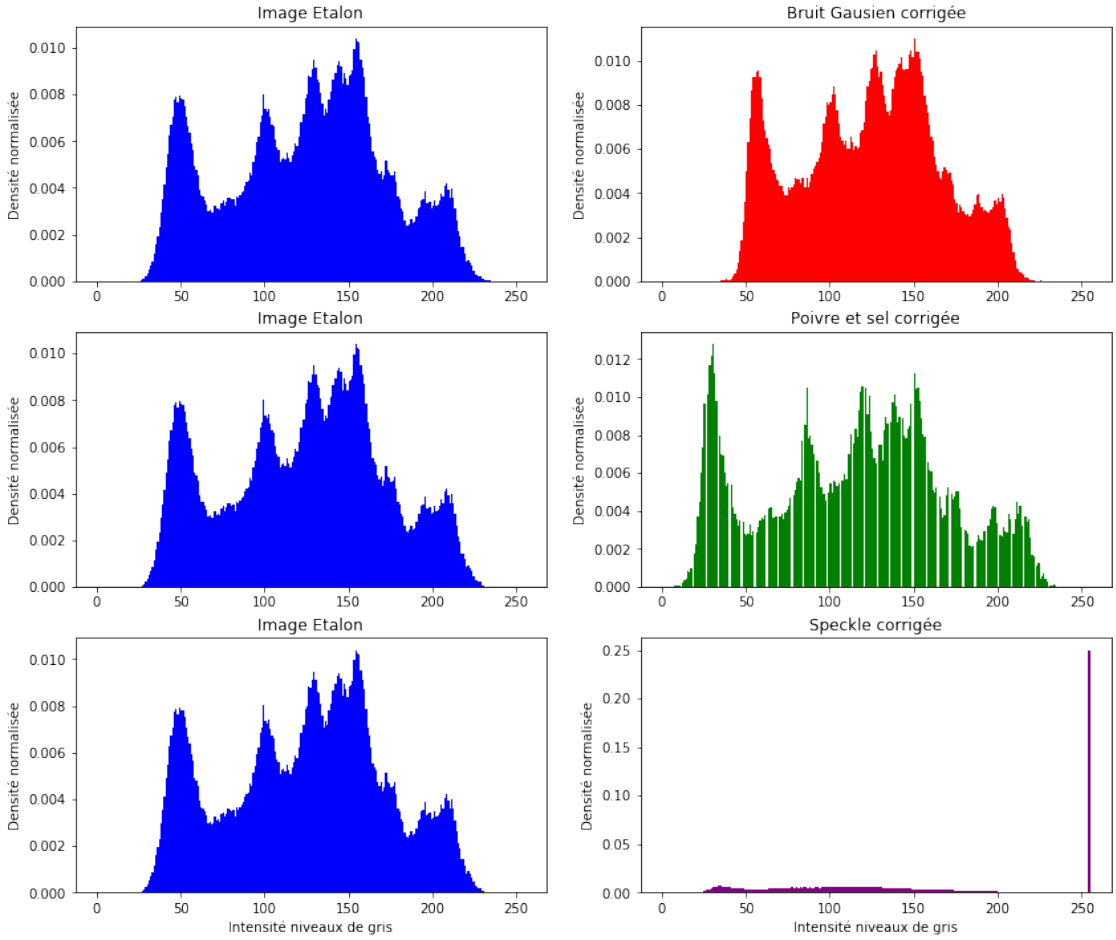
- Le filtre de type Gaussien donne de meilleurs résultats sur l'image à bruit gaussien,
- Le filtre de type médian donne de meilleurs résultats sur l'image Poivre et sel,
- Le filtre de type moyenneur donne des résultats sur l'image Speckle (à noter que la plus-value est minime néanmoins),

Chaque filtre est donc approprié selon le type d'altération à corriger sur l'image...

1.5 4. Comparaison des distributions

In [20]:
```
from PIL import Image

        # Ouverture des images les mieux corrigées
        Image_Etalon = Image.open(SAVING_DIRECTORY + '\\Lena_Etalon\\lena_original.png')
        Image_Bruit_Gaussien = Image.open(SAVING_DIRECTORY + '\\Lena_bruit_gaussien\\gaussien_
        Image_Poivre_Sel = Image.open(SAVING_DIRECTORY + '\\Lena_poivre_et_sel\\median_size_9_
        Image_Speckle = Image.open(SAVING_DIRECTORY + '\\Lena_speckle\\moyenneur_noyau_dim_5_

        # Transformation en matrice
        mat_Image_Etalon=np.array(Image_Etalon)
        mat_Image_Bruit_Gaussien=np.array(Image_Bruit_Gaussien)
        mat_Image_Poivre_Sel=np.array(Image_Poivre_Sel)
        mat_Image_Speckle=np.array(Image_Speckle)

        # Création de la grille de graphique
        fig = plt.figure(figsize=(14,12))
        ax1 = fig.add_subplot(3,2,1)
        ax2 = fig.add_subplot(3,2,2)
        ax3 = fig.add_subplot(3,2,3)
        ax4 = fig.add_subplot(3,2,4)
        ax5 = fig.add_subplot(3,2,5)
        ax6 = fig.add_subplot(3,2,6)

        # Affichage des histogrammes
        ax1.hist(mat_Image_Etalon.flatten(), bins=range(256),density=True,color='blue')
        ax3.hist(mat_Image_Etalon.flatten(), bins=range(256),density=True,color='blue')
        ax5.hist(mat_Image_Etalon.flatten(), bins=range(256),density=True,color='blue')

        ax2.hist(mat_Image_Bruit_Gaussien.flatten(), bins=range(256),density=True,color='red'
        ax4.hist(mat_Image_Poivre_Sel.flatten(), bins=range(256),density=True,color='green')
        ax6.hist(mat_Image_Speckle.flatten(), bins=range(256),density=True,color='purple')

        # Définition des légendes
        ax1.set_title('Image Etalon')
        ax1.set_ylabel('Densité normalisée')
        #ax1.set_xlabel('Intensité niveaux de gris')
```

```
ax3.set_title('Image Etalon')
ax3.set_ylabel('Densité normalisée')
#ax3.set_xlabel('Intensité niveaux de gris')

ax5.set_title('Image Etalon')
ax5.set_ylabel('Densité normalisée')
ax5.set_xlabel('Intensité niveaux de gris')

ax2.set_title('Bruit Gausien corrigée')
ax2.set_ylabel('Densité normalisée')
#ax2.set_xlabel('Intensité niveaux de gris')

ax4.set_title('Poivre et sel corrigée')
ax4.set_ylabel('Densité normalisée')
#ax4.set_xlabel('Intensité niveaux de gris')

ax6.set_title('Speckle corrigée')
ax6.set_ylabel('Densité normalisée')
ax6.set_xlabel('Intensité niveaux de gris')

# Affichage du graphique
plt.show()
```



1.5.1 On constate que les distributions des images corrigées par filtre se rapprochent de l'image étalon

1.5.2 à l'exception de l'image bruitée "Speckle" qui reste de médiocre qualité...