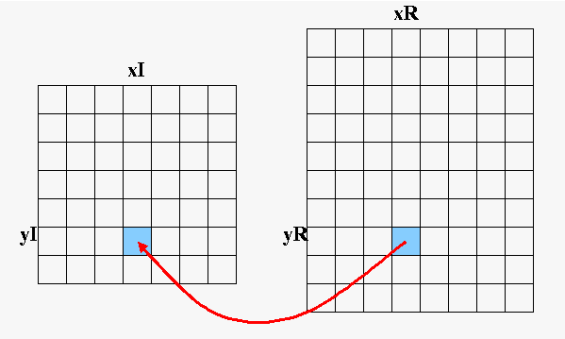
ImageJ - TP 2
Transformations géométriques

Le principe d'un algorithme de transformation géométrique appliqué à une image numérique matricielle est le suivant : pour chaque pixel de l'image résultat, on cherche le pixel correspondant dans l'image initiale (pixel antécédent).



Dans ce TP, on s'intéresse uniquement aux changements d'échelle. Le changement d'échelle est une homothétie de centre l'origine. On note sX et sY les facteurs d'échelle suivant chaque axe. Si le facteur d'échelle est plus grand que 1, on agrandit l'image, s'il est plus petit que 1 on réduit l'image. Soit (xR, yR) un pixel de l'image résultat et (xI, yI) son antécédent dans l'image initiale. On a :

$$xR = sX.xI \quad \Rightarrow \quad xI = \frac{1}{sX}.xR$$
$$yR = sY.yI \quad \Rightarrow \quad yI = \frac{1}{sY}.yR$$

L'algorithme de calcul de l'image résultat est donc le suivant :

```
wR = sX*w
hR = sY*h
créer l'image résultat iR de taille wR, hR
for(yR=0; yR<hR; yR++)
   for(xR=0; xR<hR; xR++)
      iR(xR, yR) = iI(xR/sX, yR/sY)
```

Exercice 2.1 *Agrandir/réduire une image*

1. Implémentez cet algorithme sous la forme d'un plugin ImageJ. Dans un premier temps on demandera à l'utilisateur un facteur d'échelle qui appliqué aux deux dimensions de l'image (voir ci-dessous).

```
GenericDialog gd = new GenericDialog( "Facteur d'echelle",
                                      IJ.getInstance() );
gd.addSlider( "facteur", 0.0, 10.0, 2.0 );
gd.showDialog()
if ( gd.wasCanceled() ) {
    IJ.error( "PlugIn cancelled" );
    return;
}
double ratio = (double) gd.getNextNumber();
```

   Faites en sorte que le point choisi pour l'antécédent soit le plus proche voisin du point calculé (on parle d'*interpolation au plus proche voisin*).
2. Modifier le code précédent de façon à ce que l'utilisateur fournisse la taille de l'image résultat.
3. Que remarquez vous quand vous appliquez un facteur d'échelle très supérieur à 1 ? Testez l'agrandissement d'image fait par ImageJ (`[Image/Scale]`). Avec quel type d'interpolation obtenez vous de meilleurs résultats visuels qu'avec votre code ?

Exercice 2.2 *Mosaïquage*

1. Ecrire un plugin ImageJ qui crée une nouvelle image où chaque pixel de l'image en entrée est remplacé par un carré de côté `n`, de la couleur du pixel initial. La valeur de `n` est demandée à l'utilisateur.
2. Ecrire un plugin ImageJ qui crée une nouvelle image en dupliquant l'image en entrée `nbl` fois en largeur et `nbh` fois en hauteur (les paramètres seront demandés à l'utilisateur). Testez avec l'image d'un bloc lego fournie avec le sujet.