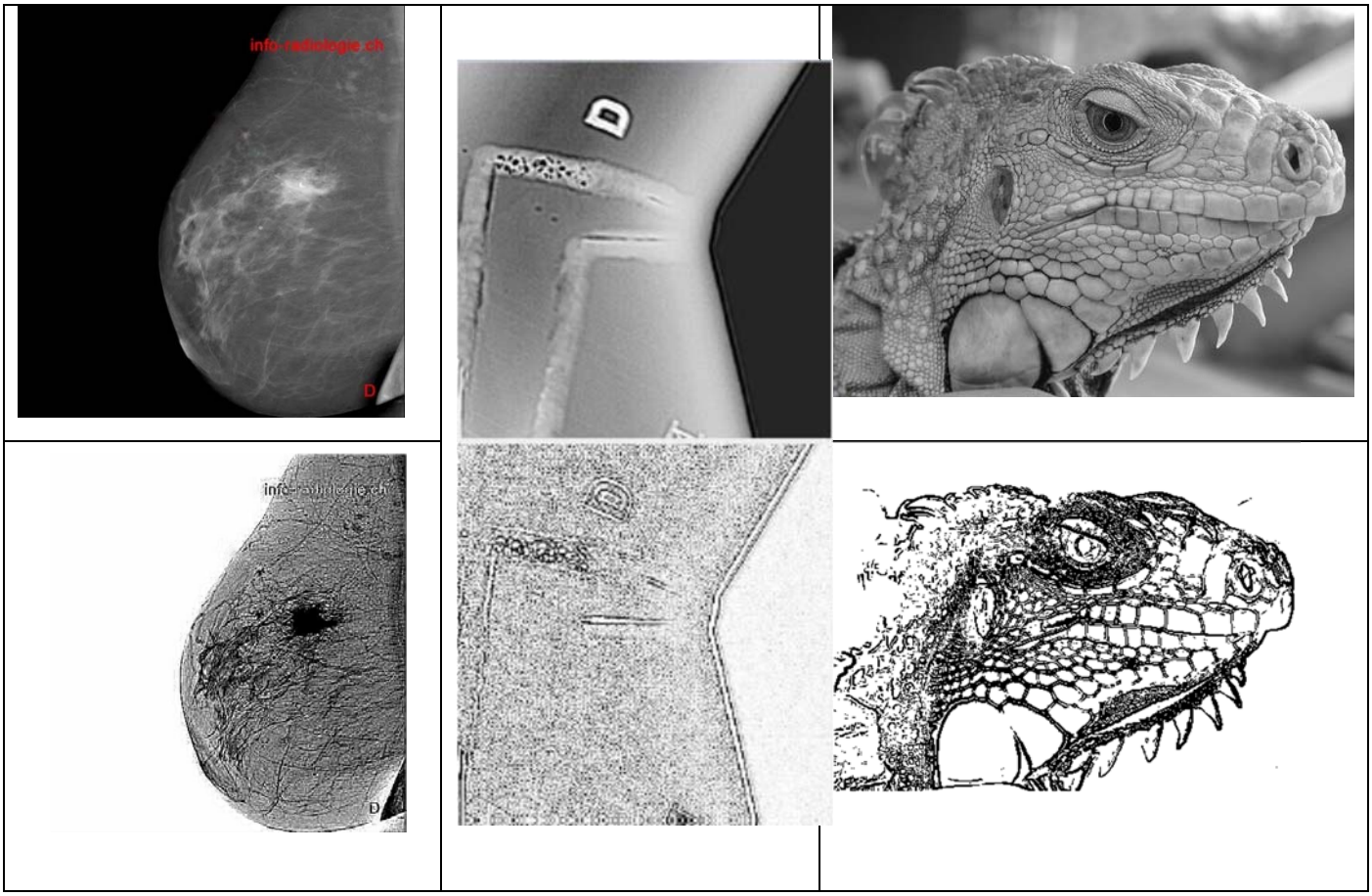
Traitement d'images : une détection de contours.

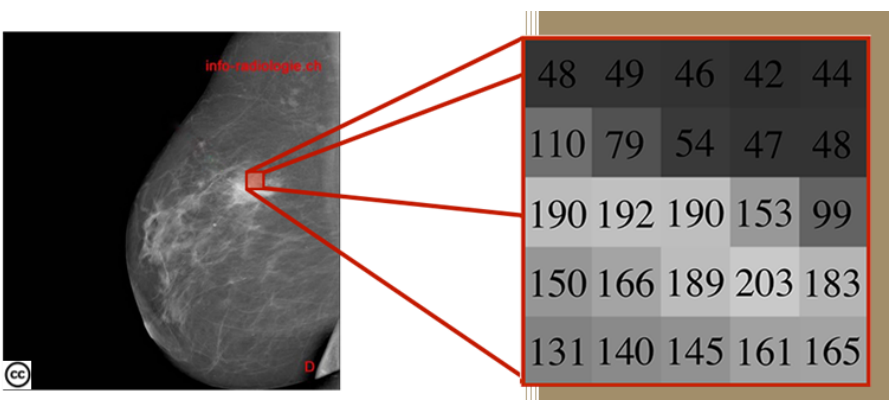
On peut souhaiter identifier les contours d'une image numérique, par exemple dans un cadre médical pour poser un diagnostic, dans un cadre industriel pour effectuer un contrôle non destructif d'une pièce ou tout simplement pour des raisons esthétiques !



Nous allons ici progressivement comprendre comment réaliser un tel traitement.

I) De l'intuition à la modélisation !

On travaille sur une image en niveaux de gris : c'est un tableau de pixels !



Question 1 : sur quel intervalle est codé le gris ?

Notre cerveau est capable rapidement, de façon non consciente, de détecter des contours.

230	200	220	210	215
212	79	54	47	195
218	54	190	153	192
220	47	49	51	183
215	200	195	192	200

Question 2 : sur ce tableau de pixels : sur quels pixels placerais-tu un contour ?

Une première modélisation : la notion de rupture de continuité

Un contour peut se caractériser par une rupture (discontinuité) d'intensité du niveau de gris dans l'image suivant une (ou des) direction(s) donnée(s).

Pour un pixel donné, si le niveau de gris des pixels voisins est "très différent" de celui de ce pixel, on considérera que ce pixel fait partie d'un contour.

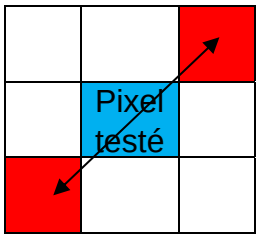
Mais comment calculer cette différence ?

Se repérer

	P(x,y)	

Question 3 : dans chacune des cases, indiquer les coordonnées du pixel en fonction de x et y.

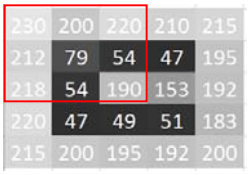
Choisir une direction



On choisit ici de comparer deux pixels situés en diagonale de part et d'autre de notre pixel central.

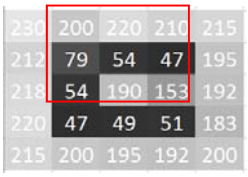
Calculer un écart

Question 4 : calculer l'écart entre les deux pixels rouges :



230	200	220
212	79	54
218	54	190

Question 5 : même question avec :



200	220	210
79	54	47
54	190	153

Pour éviter d'avoir un signe négatif dans la soustraction, il suffira de prendre "la valeur absolue" de cette différence !

Question 6 : parmi ces deux pixels et selon le critère choisi, lequel ou lesquels semble(nt) appartenir à un éventuel contour ?

II) Première automatisation avec un tableur ! La notion de seuil et de binarisation.

Pour rendre plus facile la manipulation on se propose d'utiliser une feuille d'un tableur.

Ouvrir le fichier : seuilEtape1.xlsm

Image initiale			
200	220	210	215
79	54	47	195
54	190	153	192
47	49	51	183
200	195	192	200

Avec deux voisins				
0	0	0	0	0
0	2	156	25	0
0	166	0	146	0
0	25	47	3	0
0	0	0	0	0

On remarque que les pixels du bord ont été mis à 0 : ils sont particuliers car ils ne disposent pas de 8 voisins….

Reste à définir comment nous allons décider si un pixel appartient ou non à un contour.

On décide d'utiliser un seuil : si la valeur de la différence est supérieure à ce seuil on retient le pixel !

Compléter les deux résultats obtenus pour deux valeurs du seuil :

Question 7 : seuil =20

Consigne : si la valeur de la différence est supérieure on seuil, on retient ce pixel : on mettra un 1 dans la case (0 sinon).

Question 8 : seuil = 50

On pourra vérifier les résultats à ces questions en changeant le seuil sur la feuille du tableur et en cliquant sur le bouton !

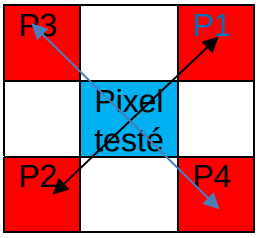
III) Généralisation à plusieurs directions !

a) Deux directions, 4 voisins !

Les résultats obtenus vont partiellement à l'encontre de notre intuition initiale

Nous n'avons en effet pris en compte que 2 pixels voisins en privilégiant une unique direction.

Pour corriger cela on va donc prendre quatre pixels.



Le problème c'est qu'une différence simple ne convient plus !

On se propose donc d'utiliser la formule suivante pour calculer cet écart entre les quatre pixels !

$$Norme = \sqrt{(Valeur\ P1 - ValeurP2)^2 + (Valeur\ P3 - Valeur\ P4)^2}$$

Question 9 : ouvre la feuille de tableur seuilEtape2.xlsm et complète les formules pour 4 voisins !

		Image initiale				Avec deux voisins					Avec 4 voisins				
230	200	220	210	215		255	255	255	255	255	255	255	255	255	255
212	79	54	47	195		255	2	156	25	255	255				255
218	54	190	153	192		255	166	0	146	255	255				255
220	47	49	51	183		255	25	47	3	255	255				255
215	200	195	192	200		255	255	255	255	255	255	255	255	255	255
						Seuil	100				Seuil	100			
		Gris				0	0	0	0	0	0	0	0	0	0
						0	0	1	0	0	0	0	0	0	0
						0	1	0	1	0	0	0	0	0	0
						0	0	0	0	0	0	0	0	0	0
						0	0	0	0	0	0	0	0	0	0

Question 10 : teste les résultats obtenus pour un seuil de 100, puis de 200 (toujours le bouton Gris).

b) Quatre directions, huit voisins !

Question 11 : ouvre la feuille de tableur seuilEtape3.xlsm et complète là pour 8 voisins (on pourra tester un seuil de 20 puis un seuil de 150).

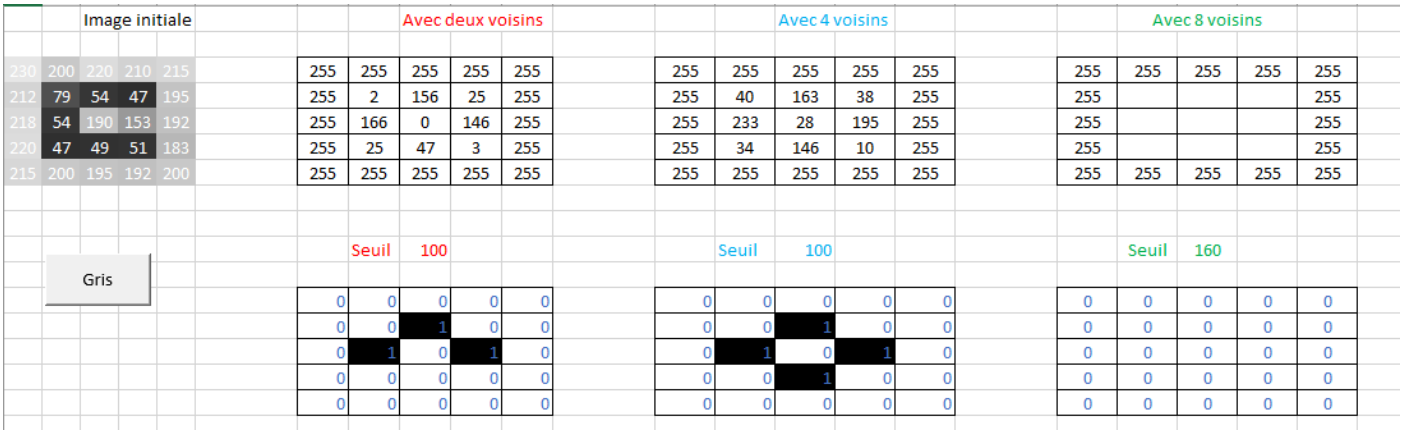


Image initiale				
230	200	220	210	215
212	79	54	47	195
218	54	190	153	192
220	47	49	51	183
215	200	195	192	200

Gris

Avec deux voisins				
255	255	255	255	255
255	2	156	25	255
255	166	0	146	255
255	25	47	3	255
255	255	255	255	255

Seuil 100

0	0	0	0	0
0	0	1	0	0
0	1	0	1	0
0	0	0	0	0
0	0	0	0	0

Avec 4 voisins				
255	255	255	255	255
255	40	163	38	255
255	233	28	195	255
255	34	146	10	255
255	255	255	255	255

Seuil 100

0	0	0	0	0
0	0	1	0	0
0	1	0	1	0
0	0	1	0	0
0	0	0	0	0

Avec 8 voisins				
255	255	255	255	255
255				255
255				255
255				255
255	255	255	255	255

Seuil 160

0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0

IV) En Python, un algorithme !

Ouvrir une image en niveaux de gris

Créer une image vide qui contiendra les contours : un pixel du contour sera noir (niveau de gris 0) sinon on le laissera blanc (niveau de gris 255).

Pour chaque pixel de l'image initiale :

Récupérer son niveau de gris

Calculer la norme sur 8 pixels voisins.

Si cette norme> seuil

Alors on met un pixel noir dans le fichier image contour

Sinon on met un pixel blanc.

En Python, nous utiliserons deux boucles for imbriquées.

Ouvrir sous Edupython le programme : DetectContour.py

Deux lignes sont à compléter : ligne 29 et 34 !

```
for  y in range(1,dimy-1) :
    for x in range ( 1, dimx-2):

        rvbCentre = image1.getpixel((x,y)) # le pixel c

        rvbVoisin1= image1.getpixel((x+1,y-1)) # Les 8
        rvbVoisin2= image1.getpixel((x-1,y+1))

        rvbVoisin3= image1.getpixel((x-1,y-1))
        rvbVoisin4= image1.getpixel((x+1,y+1))

        rvbVoisin5= image1.getpixel((x,y-1))
        rvbVoisin6= image1.getpixel((x,y+1))
        rvbVoisin7= image1.getpixel((x-1,y))
        rvbVoisin8= image1.getpixel((x+1,y))


        norme=


        if norme >      :
            image2.putpixel((x,y),(0,0,0))
```