Traitement d'image - Détection de contours

Introduction :

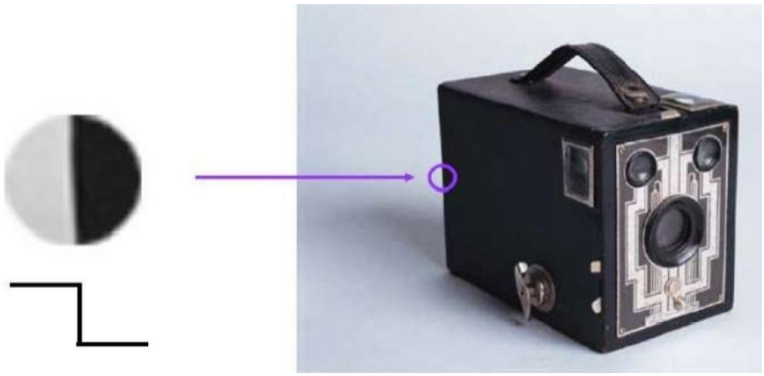
Quand une voiture autonome est en mouvement, il est important qu'elle sache reconnaître les objets qui l'entourent : véhicules, piétons, cyclistes, les panneaux de signalisation.

Pour suivre la route, elle doit savoir en particulier reconnaître les bordures, les lignes blanches, les trottoirs, etc.

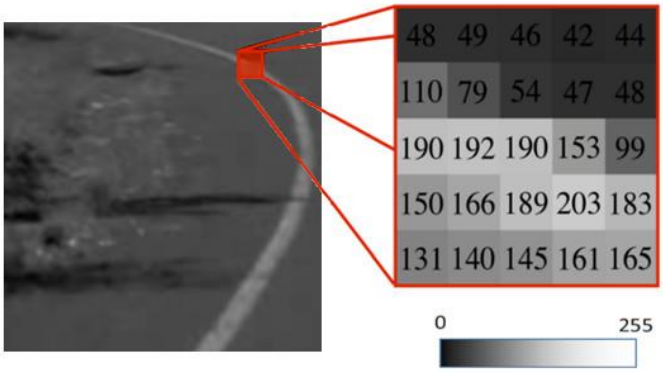
Pour cela, les photos réalisées par ses capteurs sont traitées par des algorithmes qui permettent de détecter les contours des objets qu'elle rencontre.

L'objectif de cette activité est de découvrir le fonctionnement d'un de ces algorithmes de détection de contours

Qu'est-ce qu'un contour ?



<u>Question 1</u> : Repérer et sélectionner les pixels appartenant au contour que l'on visualise sur l'agrandissement de l'image en niveaux de gris ci-dessous.



Question 2 : Expliquer comment on peut déterminer si un pixel appartient à un contour.

Pixels voisins

Pour repérer la position d'un pixel, on repère d'abord la colonne x à laquelle il appartient, puis la ligne y à laquelle il appartient.
Notons un tel pixel $\mathbf{P}(x, y)$ et $\mathbf{NG}(x, y)$ son niveau de gris.

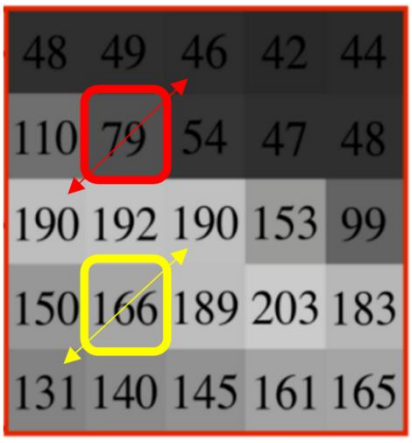
Pour détecter si le pixel $\mathbf{P}(x, y)$ appartient ou non à un contour, une des méthodes consiste à chercher une rupture d'intensité entre 2 pixels symétriques par rapport à $\mathbf{P}(x, y)$, appelés pixel voisins.

Pour cela on calcule la **différence** entre les niveaux de gris de 2 pixels voisins. Si cette différence est élevée, le pixel fera parti d'un contour. Sinon, non.

Question 3 : Compléter la position de chaque pixel composant l'image suivante et colorier d'une même couleur les couples de pixels voisins de $\mathbf{P}(x, y)$.

	$\mathbf{P}(x, y)$	
		$\mathbf{P}(x+1, y+1)$

Mesure d'une discontinuité



Question 4 : Quelle est la différence de niveaux de gris entre les 2 pixels voisins du pixel $P(2, 2)$? Ecrire l'opération effectuée.

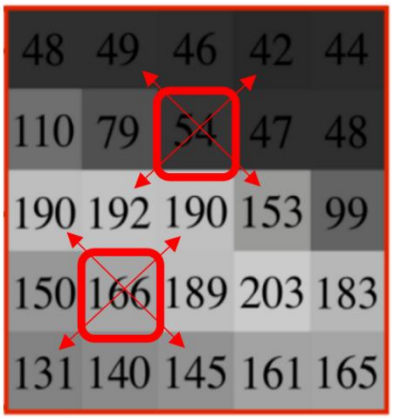
Question 5 : Même question pour le pixel $P(2, 4)$.

Question 6 : Parmi ces deux pixels, lequel semble appartenir à un éventuel contour. Pourquoi ?

Pour ne pas avoir une différence négative, il suffit d'élever au carré le résultat.

Question 7 : Reprendre les calculs précédents en élevant la différence au carré. A partir de quelle valeur (de quel seuil) pourrait-on décider qu'un pixel appartient ou non à un contour ?

Mesure d'une discontinuité avec 4 pixels voisins



Avec les exemples précédents, nous n'avons pris en compte qu'un couple de pixels voisins. On ne tient donc compte que d'une unique direction.

Pour éviter cela on va donc prendre en compte 2 couples de pixels voisins avec des directions opposés (voir schéma).

On calcule ensuite pour chacun des couples de pixels voisins, le carré de la différence des niveaux de gris. Enfin, on ajoute les résultats obtenus.

Question 8 : Calculer cette somme pour $\mathbf{P(3, 2)}$ et $\mathbf{P(2, 4)}$.

Question 9 : Exprimer cette somme pour le pixel $\mathbf{P(x, y)}$.

Notion de seuil de décision

Question 10 : A partir que quel seuil peut-on décider si un pixel fait partie du contour ?

Programmation en Python de la détection de contours d'une image

On a programmé en langage Python, un algorithme qui permet d'afficher le contour d'une image.

Question 11 : Lire attentivement le programme python suivant et compléter la ligne 22.

```
1  from PIL import Image
2
3  # on crée une fonction contour pour détourer une image en niveaux de gris.
4  # la variable sera le nom du fichier, à inscrire entre 'nomdufichier.png'
5  def contour(filename):
6      # on ouvre l'image qu'on va détourer. Dans la suite l'image sera appelée im
7      im = Image.open(filename)
8
9      # on crée une nouvelle image dans laquelle on créera le contour de im
10     im_contour = Image.new(im.mode, im.size)
11     colonne, ligne = im.size
12     # la double boucle for suivante va permettre de balayer les pixels de im
13     for y in range(1, ligne-1):
14         for x in range(1, colonne-1):
15             # a, b, c et d contiennent les valeurs des pixels voisins du pixel P(x,y)
16             a = im.getpixel((x-1, y-1))
17             b = im.getpixel((x+1, y+1))
18             c = im.getpixel((x-1, y+1))
19             d = im.getpixel((x+1, y-1))
20
21             # c'est la ligne que vous devez compléter.
22             formule =
23
24             # on construit un test : si la valeur obtenue dans la formule est
25             # supérieure au seuil de 2000, le pixel (x,y) fait parti du contour,
26             # on le colorie en noir. Sinon, on le colorie en blanc.
27             if formule > 2000:
28                 im_contour.putpixel((x, y), 0)
29             else :
30                 im_contour.putpixel((x, y), 255)
31
32     #l'instruction putpixel permet de remplacer la valeur d'un pixel(x,y)par une autre.
33     im_contour.show()
```