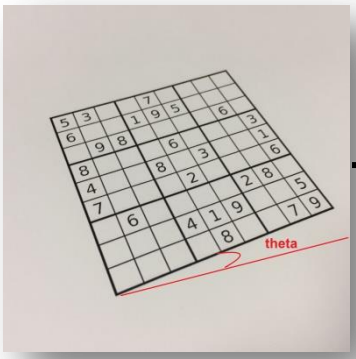
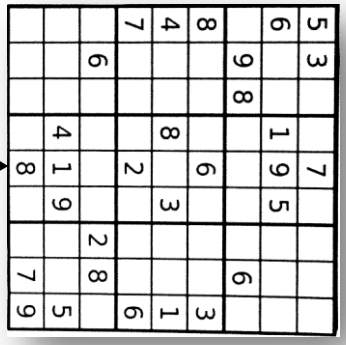


La photo prise par le Raspberry Pi est filtrée pour éliminer le bruit. Le programme thresholding.py permet de récupérer une image droite de la grille en se servant de techniques de détection de contours. L'angle de rotation de la grille est aussi retourné.



L'image récupérée peut être inversée il faut alors détecter l'inversion avant de pouvoir détecter la grille. Pour se faire l'image reçue est rotée 4 fois et à chaque fois on calcule la cohérence de la détection (« sorte de corrélation ») et on ne garde que la rotation la plus cohérente

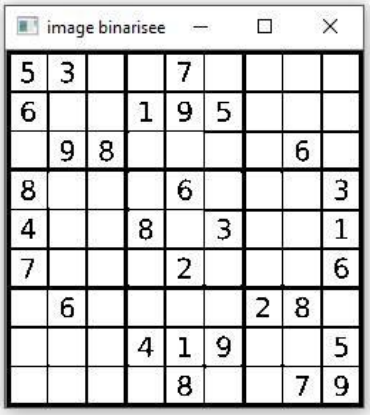
5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

```
>>>
 RESTART: C:\Users\Mohamed\Documents\2ASicom\Sudoku-robot\Firmware\mnist_model_
convolutionnel.py
opencv function connectedComponentsWithStats() not found or maybe only one argu
ment is given.

... findContours() method will be used!

Extracting /tmp/tensorflow\train-images-idx3-ubyte.gz
Extracting /tmp/tensorflow\train-labels-idx1-ubyte.gz
Extracting /tmp/tensorflow\t10k-images-idx3-ubyte.gz
Extracting /tmp/tensorflow\t10k-labels-idx1-ubyte.gz
l'image est tournée de  90
>>> recogn
array([[5, 3, 0, 0, 7, 0, 0, 0, 0],
       [6, 0, 0, 1, 9, 5, 0, 0, 0],
       [0, 9, 8, 0, 0, 0, 0, 6, 0],
       [8, 0, 0, 0, 6, 0, 0, 0, 3],
       [4, 0, 0, 8, 0, 3, 0, 0, 1],
       [7, 0, 0, 0, 2, 0, 0, 0, 6],
       [0, 6, 0, 0, 0, 0, 2, 8, 0],
       [0, 0, 0, 1, 1, 9, 0, 0, 5],
       [0, 0, 0, 0, 8, 0, 0, 7, 9]], dtype=int64)
```

Une fois la rotation éventuelle de la grille détectée et corrigée on peut maintenant reconnaitre les chiffres par des techniques de réseaux de neurone ici convolutifs. Nous avons constitué une base de données d'entrainement de 850 images avec une base de test de 150 images.

On a une précision de 96% . Le programme processing.py + mnist_model_convolutionnel.py reconnait à la fois les cases vides et remplies

```
>>> cv2.imshow('image binarisee',imB)
>>> recogn
array([[5, 3, 0, 0, 7, 0, 0, 0, 0],
       [6, 0, 0, 1, 9, 5, 0, 0, 0],
       [0, 9, 8, 0, 0, 0, 0, 6, 0],
       [8, 0, 0, 0, 6, 0, 0, 0, 3],
       [4, 0, 0, 8, 0, 3, 0, 0, 1],
       [7, 0, 0, 0, 2, 0, 0, 0, 6],
       [0, 6, 0, 0, 0, 0, 2, 8, 0],
       [0, 0, 0, 4, 1, 9, 0, 0, 5],
       [0, 0, 0, 0, 8, 0, 0, 7, 9]], dtype=int64)
>>> np.array(resolver(t))
array([[5, 3, 4, 6, 7, 8, 9, 1, 2],
       [6, 7, 2, 1, 9, 5, 3, 4, 8],
       [1, 9, 8, 3, 4, 2, 5, 6, 7],
       [8, 5, 9, 7, 6, 1, 4, 2, 3],
       [4, 2, 6, 8, 5, 3, 7, 9, 1],
       [7, 1, 3, 9, 2, 4, 8, 5, 6],
       [9, 6, 1, 5, 3, 7, 2, 8, 4],
       [2, 8, 7, 4, 1, 9, 6, 3, 5],
       [3, 4, 5, 2, 8, 6, 1, 7, 9]])
>>>
```



Le sudoku reconstitué, on peut maintenant le résoudre : programme resolver.py

Il ne reste plus qu'à commander les moteurs. Fait par le programme control_motor.py

Ce qui reste à faire : Mettre en place la chaine, faire les quelques réglages qui s'imposent pour la partie mécanique puis valider tous les programmes avant de passer à l'asservissement vidéo.

Code à retrouver sur github : https://github.com/Sanahm/Sudoku-robot/

BERTRAND Emile

SANA Mohamed

BOUDIER Baptiste

GENTIL Kévin

Tuteur : Bertrand RIVET