L3 - *« Computer Vision »* *L. Mascarilla, B. Besserer*	**TP fonctionnement «Headless» - Transformée de Hough**	La Rochelle Université

L'objectif de ce TP est de créer une application qui utilisera la JeVois en mode autonome, c'est-à-dire sans qu'elle soit reliée à un ordinateur. Il s'agit de réaliser un niveau à bulle grâce à la transformée de Hough.

A la fin du TP merci d'effacer votre module afin que le groupe suivant puisse travailler dans de bonnes conditions.

Détection de lignes horizontales

A) Version 1 : Développement sur PC

Vous utiliserez comme point de départ le tutoriel d'Opencv sur la transformée classique de **Hough** qui est implémentée sous le nom **HoughLines :**

Tutoriel Hough Lines

Vous adapterez le code en :

- utilisant comme entrée la webcam de votre ordinateur (et non une image de Sudoku!).

- traçant uniquement les lignes proches de l'horizontale avec la convention suivante :

    - en rouge les lignes dont l'angle est supérieur à l'horizontale

    - en bleu celles d'angle inférieur

    - en vert celles qui sont «proches » de l'horizontale.

Pour sélectionner les lignes horizontales, vous avez deux possibilités :

a) limiter l'intervalle des angles pris en compte par HoughLines() autour de l'horizontale :

angle_horizontal ± eps

b) tester la valeur des angles donnés par HoughLines et ne tracer que ceux qui vous intéressent.

Je vous suggère de combiner les deux approches : la méthode a) vous permet d'éliminer rapidement les lignes « trop peu horizontales », b) permet de choisir la couleur du tracé.

Remarques :

- Pour sélectionner les lignes horizontales, vous pouvez utiliser la fonction `math.isclose()` : le test d'égalité stricte entre réels n'a le plus souvent pas beaucoup de sens, de plus vous pouvez, avec cette méthode, fixer la sensibilité de votre détecteur.

- Vous pouvez améliorer le résultat de la segmentation (avant d'appliquer le détecteur de Hough) en :

    - lissant l'image en niveau de gris par **cv2.GaussianBlur(gray, (3, 3),0)**. Si la taille du noyau et augmentée vous aurez moins de bruit mais peut être moins de lignes détectées.

    - remplacant la fonction **cv2.Canny** par la fonction suivante qui estime automatiquement

les « meilleurs » paramètres pour la segmentation :

```
def auto_canny(gray, sigma=0.33):
    # https://pyimagesearch.com/2015/04/06/zero-parameter-automatic-canny-edge-detection-with-python-and-opencv/
    # compute the median of the single channel pixel intensities
    v = np.median(gray .ravel())
    # apply automatic Canny edge detection using the computed median
    lower = int(max(0, (1.0 - sigma) * v))
    upper = int(min(255, (1.0 + sigma) * v))
    edged = cv2.Canny(gray, lower, upper)
    # return the edged image
    return edged
```

- Dans le tutoriel, une autre fonction HoughLinesP est également utilisée, vous pouvez regarder ce qu'elle apporte par rapport à HoughLines mais elle n'est pas forcément utile dans notre cas.

B) Version 2 : toujours sur PC

Comme vous pouvez le constater, la méthode précédente est souvent trop sensible (trop de lignes détectées, instabilité…). Pour l'améliorer, vous allez modifier votre code pour qu'il affiche uniquement la direction moyenne des lignes proches de l'horizontale :

- calculer l'angle moyen des lignes (« horizontales ») détectées. Pour une série d'angles mesurés $\alpha_1, \ldots, \alpha_n$ la moyenne $\bar{\alpha}$ peut être obtenue en utilisant la formule suivante (voir Wikipédia https://en.wikipedia.org/wiki/Circular_mean) :

$$\bar{\alpha} = \operatorname{atan2}\left(\sum_{j=1}^{n} \sin \alpha_j, \sum_{j=1}^{n} \cos \alpha_j\right)$$

- tracez au milieu de l'image une seule ligne, avec la convention de couleur précédente, qui a comme angle cet angle moyen.

Quand votre code fonctionne, écrivez une fonction niveau qui prend en entrée l'image à analyser (la trame courante), effectue les mêmes traitements que précédemment mais n'effectue aucun tracé et qui retourne 4 paramètres :

- **niv** :0 si l'angle moyen est supérieur à l'horizontale, 1 s'il est horizontal, 2 sinon, None s'il n'y a pas de détection.

- **theta_mean** : l'angle moyen

- **lines** : les lignes extraites par Hough

- **seg** : l'image résultat de la segmentation par le détecteur de Canny.

Tous les tracés sont effectués dans la boucle de lecture de la video à partir de ces résultats. Adaptez votre code et testez cette fonction qui nous servira dans la partie suivante.

C) Version 3 : passage sur la JeVois

Créez un module sur la JeVois dans lequel vous allez porter le code précédent. Vous utiliserez l'option « Minimal OpenCv Module » lors de sa création.

Vous imposerez que ce module soit le module pas défaut en ne plaçant qu'une seule ligne dans le fichier **videomappings.cfg**, par exemple :

```
YUYV 320 240 20 YUYV 320 240 30 LM Hough *
```

Dans un premier temps, vous allez tester votre code en visualisant les résultats sur ordinateur puis , dans un second temps, en mode « headless », c'est-à-dire sans affichage et sans liaison avec l'ordinateur.

a) Fonctionnement normal, connecté sur USB, avec JeVois Inventor

La configuration est la suivante :

- La Camera JeVois est relié à l'ordinateur

- L'afficheur 7 segments est relié à la liaison série (connecteur 4 broches) de la JeVois

- L'afficheur 7 segments est alimenté par la PowerBank.

Voir la figure suivante.

Attention : la consommation électrique de l'afficheur seul est très faible et la PowerBank se met en veille après quelques dizaines de secondes. Appuyez sur le bouton de la PowerBank pour relancer.

Pour pouvoir utiliser la liaison série avec l'afficheur 7 segments, il faut modifier la configuration de cette liaison :

- monter la SDCard de la JeVois comme un disque dur : onglet **System** puis cliquez sur **Enable** (Export microSD card inside JeVois to host computer)

- placer les informations suivantes dans le fichier **params.cfg** du répertoire **config** :

serialdev=/dev/ttyS0

serial:baudrate=9600

serial:linestyle=Zero

Remarque: le fichier params.cfg à modifier est bien celui du répertoire **config** directement sous la racine de la microSD, **il ne faut pas utiliser le fichier params.cfg de l'onglet Config qui contient le paramétrage du module courant.**

- **Redémarrez** la JeVois.

- Dans l'onglet **Console**, il faut rediriger l'écriture sur l'interface série hardware : enfoncez le bouton « 4-pin » à droite (coté « module output »). La caméra est prête à être utilisée.

1) Portez votre code dans la méthode process qui prend une image en entrée et une image en sortie :

```
def process(self, inframe, outframe)
```

2) Complétez le code pour écrire le résultat de la détection sur l'afficheur. Pour cela, incluez les instructions fournies dans le fichier « Seg7.py » puis vous utiliserez les instructions suivantes :

- **jevois.sendSerial(FormatSevenSegString("oui"))** qui affiche « oui ».

Seuls 3 caractères peuvent être affichés simultanément et tous les caractères ne peuvent être représentés par un afficheur 7 segments. Un caractère non affichable est remplacé par un tiret « - ».

Exemples de chaines valides :

str = "oui"

str = "off"

str = "up"

- **jevois.sendSerial(FormatSevenSegOff())** qui éteint l'afficheur.

Dans le cas qui nous intéresse, je vous suggère d'utiliser les chaines suivantes :

- "zzz" si aucune ligne n'est détectée

- "zoz" si la ligne moyenne est horizontale

- "ozz" si la ligne penche vers le bas

-"zzo" sinon

Le « o » représente la bulle dans un niveau, les « z » sont non affichables et donc remplacés par des tirets « - ». Pour choisir le sens de déplacement de la bulle, n'oubliez pas de regarder le placement de la JeVois sur la PowerBank !

Ces chaines de caractères pourront être fournies par une version modifiée de la fonction **niveau** écrite précédemment.

b) Fonctionnement en mode "headless" (fonctionnement indépendant de l'ordinateur)

Dans ce mode, la JeVois n'utilise plus la méthode **process** mais la méthode :

```
def processNoUSB(self, inframe):
```

Remarque : Vous pouvez laisser inchangée la méthode process, qui sera utilisée en mode connecté.

Pour l'essentiel, il s'agit de reprendre le code de la méthode process, mais dans lequel toute réference à l'image de sortie est supprimée (elle ne fait de toute façon pas partie des paramètres de processNoUSB).
Essentiellement, cette méthode récupérera la chaine de caractères ch fournie par la fonction niveau et l'affichera par jevois.sendSerial(FormatSevenSegString(ch)).

Quand cela est fait, il reste une dernière manipulation pour fonctionner en mode headless :

Dans l'onglet **Config**, prendre l'entrée **initscript.cfg** et y mettre les informations suivantes en remplaçant XXXVendor et XXXModuleName par la valeur que vous avez choisie pour le Vendor et le nom du module (attention, ces noms sont sensibles à la casse).

Il faut aussi adapter codage, format et framerate en fonction de ce qui est utilisé par votre module (ce sont les mêmes informations que dans videomappings.cfg).

setmapping2 YUYV 320 240 20 XXXVendor XXXModuleName

setpar serlog USB

setpar serout Hard

streamon

start

Explications :
- serpar serlog USB redirige les informations de log vers la liaison USB.

- setpar serout Hard redirige les informations écrites sur la liaison série vers l'interface hardware (connecteur 4 fils)

En sauvegardant les informations de configuration, la caméra jeVois reboote et elle doit être capable de fonctionner indépendamment de l'ordinateur : c'est-à-dire alimentée par la PowerBank, avec uniquement l'afficheur 7 segments comme sortie. Vous pouvez débrancher la prise USB de la JeVois et la relier à la PowerBank.

Il ne vous reste plus qu'à tester votre niveau à bulle !