Poppy ErgoJr réel : Mouvements et camera

Découvrez pas à pas la programmation du robot ErgoJr, ses mouvements, sa camera et l'utilisation d'OpenCV pour détecter des visages dans une image.

Sommaire

LE PRINCIPE EN VIDÉO

Voici une vidéo qui parcourt rapidement l'ensemble du sujet (mouvements, camera et détection du visage...) :

DÉMARRAGE :

Ouvrez Jupyter-NoteBook dans l'interface du robot (poppy.local).

Pour exécuter une cellule, vous pouvez cliquer sur l'icône "Play" ou appuyer sur Ctrl+Entrée.

Il faut commencer par importer les librairies du robot et définir un robot.

Ici, nous l'appellerons "poppy".

Cette opération doit être effectuée à chaque nouvelle prise en main du robot (chaque démarrage).

```
1. from poppy.creatures import PoppyErgoJr
2. poppy = PoppyErgoJr()
```

Télécharger

L'activation des moteurs nécessite de fixer au préalable leurs mouvements afin de ne pas se coincer les doigts par exemple.

Après cette opération, les moteurs ne peuvent plus bouger à la main.

```
#Une boucle sur les six moteurs :
for m in poppy.motors:
    m.compliant = False #Rigidite des moteurs
```

Télécharger

Une fois ces opérations effectuées, vous pouvez commencer à actionner les moteurs...

MOUVEMENTS DU ROBOT :

Un simple mouvement du premier moteur :

```
poppy.m1.goto_position(-30, 0.5, wait=True)
poppy.m1.goto_position(30, 1, wait=True)
```

Télécharger

- 30 ou -30 correspond à l'angle final demandé
- 0.5 ou 1 correspond à la durée totale du mouvement (plus ou moins rapide)
- *wait=True* permet d'attendre la fin du mouvement avant de commencer le suivant.

En passant le paramètre *wait* à *False*, on peut faire bouger plusieurs moteurs en même temps :

```
poppy.m2.goto_position(-50, 0.5, wait=False)
poppy.m3.goto_position(50, 0.5, wait=True)
```

Télécharger

Enfin, on peut créer mémoriser une position (initiale par exemple) par une liste de positions sur chaque moteur.

Les moteurs sont actionnés en même temps.

Le paramètre *wait=True* permet d'attendre la fin de la totalité du mouvement avant de passer à un autre.

```
#Position initiale :
pos_1 = {'m1': 0, 'm2': -90, 'm3': 70, 'm4': 0, 'm5': -70, 'm6': 50}
poppy.goto_position(pos_1, 1, wait=True)
```

Télécharger

LES LEDS :

Chaque moteur est équipé de LEDs. Il est donc possible de les allumer avec des couleurs différentes :

```
poppy.m1.led = 'red' #La LED du moteur 1 passe au rouge
```

```
poppy.m2.led = 'green' #La LED du moteur 2 passe au vert
```

Il faut aussi pouvoir les éteindre :

```
#Avec une boucle sur tous les moteurs par exemple :
for m in poppy.motors:
    m.led = 'off' #On eteint les LEDs
```

Télécharger

CAMERA ET PHOTOS :

Poppy est équipé d'une camera. Cela permet d'enregistrer des vidéos, de prendre des photos... **C'est bien mais que peut-on en faire ???**

Dans les librairies Python présentes sur poppy, il en existe une particulièrement intéressante : **OpenCV**

C'est une librairie de traitements et d'analyses d'images.

Grâce à des modules déjà construits, il est possible de détecter des formes, des visages, des couleurs... et plein d'autres choses...

Prenons juste une photo (pour l'instant) :

```
#Import des librairies :
%pylab inline
import cv2
import matplotlib.pyplot as plt

#Prise de la photo :
image = poppy.camera.frame

#Affichage de la photo :
plt.imshow(cv2.cvtColor(image,cv2.COLOR_BGR2RGB))
```

Télécharger

Résultat :

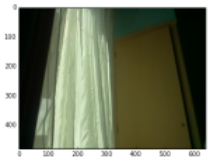


photo1 poppy
Une photo prise par poppy ErgoJr

DÉTECTION D'UN VISAGE :

Pour cela, il faut charger un fichier de configurations puis observer les contours des objets dans l'image en noir et blanc (échelle de gris).

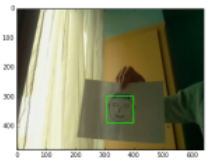
Un exemple tout de suite. J'ai accepté de mettre ma tête sur le web pour ce tuto.

Quelques remarques utiles viennent après le code :

```
%pylab inline
import cv2
import matplotlib.pyplot as plt

#Chargement du fichier de configurations:
chemin='/home/poppy/miniconda/share/OpenCV/haarcascades/haarcascade_frontalface_default.xml'
faceCascade = cv2.CascadeClassifier(chemin)

#Prise de la photo:
image = poppy.camera.frame

#Passer en niveaux de gris :
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

# Detection des visages dans l'image:
faces = faceCascade.detectMultiScale(
    gray,
    scaleFactor=1.1,
    minNeighbors=5,
    minSize=(70, 70),
    flags = cv2.CASCADE_SCALE_IMAGE
    )

#On print le nombre de visages detectes dans l'image:
print "{0} visage(s) trouve(s)!".format(len(faces))

#On dessine un rectangle sur l'image pour chaque visage detecte:
for (x, y, w, h) in faces:
    cv2.rectangle(image, (x, y), (x+w, y+h), (0, 255, 0), 2)

#On affiche l'image:
plt.imshow(cv2.cvtColor(image,cv2.COLOR_BGR2RGB))
```

Télécharger

Résultat : 1 visage(s) trouve(s) !



poppy visage1
Une détection d'un visage grâce à openCV

Remarques et erreurs éventuelles :

- Le fichier xml de configurations est situé ici sur poppy :

/home/poppy/miniconda/share/OpenCV/haarcascades/haarcascade_frontalface_default.xml

Si le chemin n'est pas bon, votre programme va charger un fichier vide ce qui aboutira à l'erreur suivante :

error : (-215) !empty() in function detectMultiScale

- Certains facteurs engendrent des détections érronées (lumière, vétements, autres objets...).

Il faut alors jouer sur certains paramètres de la fonction *detectMultiScale()* :

Modifier le *scaleFactor* ou augmenter/diminuer la taille minimale de la détection : *minSize= (50,50)*...

RÉACTION À LA DÉTECTION DE VISAGES

Dans l'exemple suivant, poppy prend une photo toutes les deux secondes.

S'il détecte un visage, les LEDs passent au vert, il se lève un peu puis dit "Bonjour !" en hochant la tête. Les LEDs s'éteignent, on sort de la boucle, on affiche la dernière image puis le programme s'arrête.

Sinon, s'il ne détecte pas de visage, il remue la te de gauche à droite (pour chercher) puis reprend une autre photo.

(*Voir la vidéo d'introduction)*

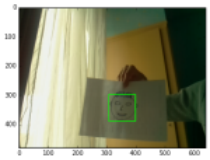
```
%pylab inline
#import numpy as np

import cv2
import time

import matplotlib.pyplot as plt
#import pypot

for m in poppy.motors:
    m.compliant = False
    m.led = 'red'

#Position initiale :
pos_1 = {'m1': 0, 'm2': -90, 'm3': 70, 'm4': 0, 'm5': -70, 'm6': 50}
poppy.goto_position(pos_1, 1, wait=True)

#Chargement du fichier de configurations:
chemin='/home/poppy/miniconda/share/OpenCV/haarcascades/haarcascade_frontalface_default.xml'
faceCascade = cv2.CascadeClassifier(chemin)

#Boucle principale:
Play=True
while Play :
    #Prise de la photo:
    image = poppy.camera.frame

    #Passer en niveaux de gris :
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

    #Detection des visages dans l'image:
    faces = faceCascade.detectMultiScale(
        gray,
        scaleFactor=1.1,
        minNeighbors=5,
        minSize=(70, 70),
        flags = cv2.CASCADE_SCALE_IMAGE
    )

    #On print le nombre de visages detectes dans l'image:
    print "{0} visage(s) trouve(s)!".format(len(faces))

    #On dessine un rectangle sur l'image pour chaque visage detecte:
    for (x, y, w, h) in faces:
```

```
        cv2.rectangle(image, (x, y), (x+w, y+h), (0, 255, 0), 2)

    #Si au moins un visage est detecte, les LEDs passent au vert,
    #poppy se tourne vers le visage et dit "Bonjour!":
    if len(faces)>0:
        #Les LED passent au vert :
        for m in poppy.motors:
            m.led = 'green'

        #Poppy se tourne vers le visage:
        #(La largeur de l'image est de 640 px.
        #On adapte alors une fonction pour produire des angles de -30 à 30)
        x=faces[0][0]
        w=faces[0][2]
        poppy.m1.goto_position(30-int(60*(x+w/2)/640), 1, wait=True)

        #Poppy dit Bonjour!: (Il se leve un peu et secoue la tete)
        poppy.m3.goto_position(20, 1, wait=False)
        poppy.m2.goto_position(-20, 1, wait=True)
        a=30
        for i in range(0,5):
            a=-1*a
            poppy.m6.goto_position(50+a, 0.5, wait=True)
        poppy.m6.goto_position(50, 1, wait=True)

        #Enfin, On eteint les lED:
        for m in poppy.motors:
            m.led = 'off'
        #On arrete la boucle... ou pas...
        Play=False

    #Si pas de visage detecte, poppy bouge sa tete de gauche a droite pour chercher:
    else:
        #Poppy dit cherche:
        a=30
        for i in range(0,2):
            a=-1*a
            poppy.m1.goto_position(a, 1, wait=True)

    #On attend avant une nouvelle photo:
    time.sleep(2)

#Une fois la boucle terminee, on affiche la dernière image pour verifier:
plt.imshow(cv2.cvtColor(image,cv2.COLOR_BGR2RGB))
```

Télécharger

Résultat :

```
0 visage(s) trouve(s) !
0 visage(s) trouve(s) !
0 visage(s) trouve(s) !
0 visage(s) trouve(s) !
1 visage(s) trouve(s) !
```



poppy visage1
Une détection d'un visage grâce à openCV

TÉLÉCHARGEMENTS :

Voici le notebook complet (dans un zip) :

iPython NoteBook : Detection d'un visage
L'article au complet

Un lien vers le contenu du NoteBook en format *html* :

NoteBook en format html
Le contenu de l'article.

samedi 10 septembre 2016 par wlaidet