Lab Notebook MSC

Baptiste Rouger

31 janvier 2018

Table des matières

15 Jan 2018

- Début d'installation sur le PC
- Cassage de ArchLinux
- Rangement de la salle de manip et mise en place de la salle de manip
  - Raccourcissement des plus longues barres de la cage qui gênaient.
  - Réorganisation de la salle
- Lecture de la review de Mathieu

Comment la température ambiante influence la fermeture des feuilles ?
Comment la fermeture des feuilles affecte la température de celles-ci ?

16 Jan 2018

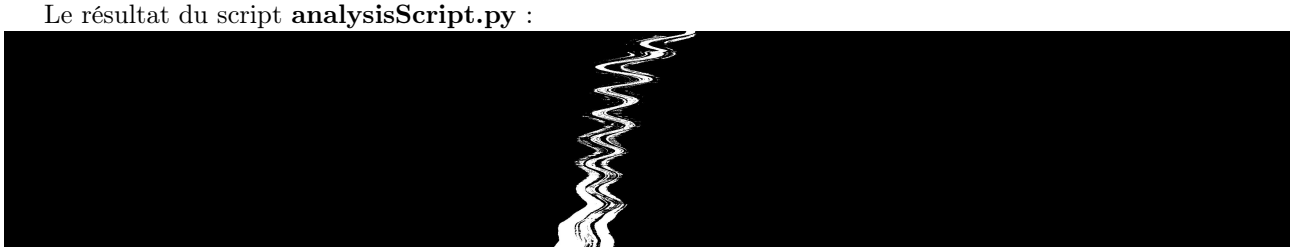
- Installation de Debian sur le PC
- Installation des logiciels importants sur le PC
- Mise en place de la première manip test pour la **nutation** : début à **15h52**, fin à **10h42** le 17 Jan 2018. Les données sont situées sur Alfred : /mnt/data/manip/Baptiste/test_16-01-2018.J'ai utilisé la plante "Abby" pour réaliser cette manip. Photo toutes les 90 secondes.

17 Jan 2018

- Arrêt de la manip **test_16-01-2018** à **10h42**
- Arrosage des plantes
- Récupération des données de la manip
- Création du film à partir des données de la manip (*cf* Protocole)
- Réalisation du script **analysisScript.py** qui, à partir de photos stockées dans un dossier, réalise la timeline d'une ligne de pixel et la converti en image binaire

Le lien pour la vidéo : `http://uptobox.com/5x80eimcd7xu`

Le résultat du script **analysisScript.py** :

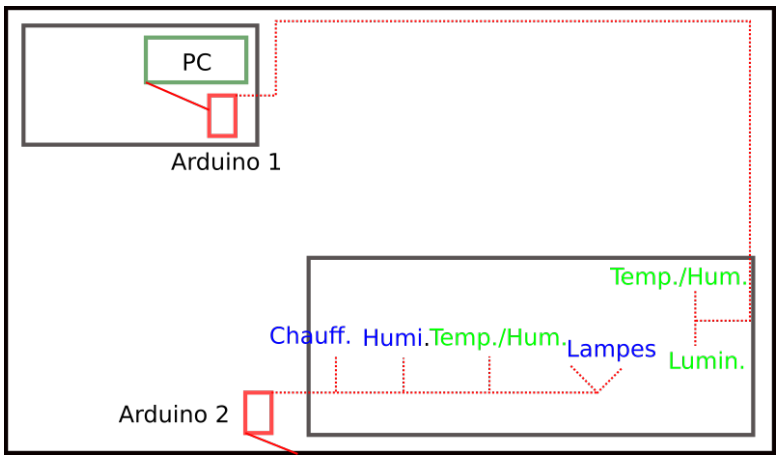


18 Jan 2018

- Début d'une deuxième manip pour observer la nutation (sur un plus long terme). Début à environ 12h30 **(heure pc décalée ? ? ? ?)**. Il fait 24.2 °C dans la pièce, 49% d'humidité. On utilise la plante "Béa". Les numéro de protocole sont :
  - arduino : 21608
  - photos : 21821
- Création du script appelé **anSc3.py** qui utilise la librairie skimage pour analyser les images et réaliser le graphe de mouvement.

19 Jan 2018

Je me suis penché sur le problème du maintien de la température et de l'humidité de la salle de manip avec Arduino (ainsi que la photopériode). J'ai pris en compte la mesure de ces données. Schéma :



20 Jan 2018

J'ai refait le focus sur la jeune feuille, il n'était évidemment plus bon au bout de ces 2 jours et demi. ⇒ Il serait intéressant de voir si l'auto focus pourrait compenser intelligemment cela.

Pour les scripts, je vais essayer de trouver la position du centre de la tige en utilisant une moyenne pondérée par le nombre de pixels consécutifs

J'ai modifié les scripts : je les ai séparé en deux : `TL.py` sert à créer l'image de la timeline, `Trajec.py` sert à récupérer la position moyenne de la tige. On peut entre les deux scripts retailler l'image.

29 Jan 2018

J'ai travaillé sur les scripts d'analyse de l'image, de récupération de la trajectoire et d'analyse en ondelette de ces trajectoires. J'ai découpé chaque étape et je les ai mises dans des scripts précédents qui prennent différents arguments (*cf* Protocole).
J'ai fini le robot arduino qui contrôle la température, l'humidité et les lampes. Je le teste pour la nuit.

30 Jan 2018

Il y avait un problème dans le robot de contôle. J'ai dû modifier le code d'une part (problème de LOW et HIGH en output pour les relais). De plus, j'ai dû installer la librairie Time sur le pc de la salle de manip. J'ai vérifié que cela fonctionnait bien quelques minutes. Le nouveau code arduino pour le robot est joint dans le dépôt : `arduino_Stat.ino`.

J'ai remarqué que la lampe numéro 2 (au dessus de la cage) ne fonctionne plus.

31 Jan 2018

Le robot de contrôle était encore bugé. Il se trouve que les relais se bloquent parfois et restent en position de contact alors qu'ils devraient couper le courant. Cela n'arrive que quand on branche la grosse lampe. Peut-être tire-t-elle trop de courant pour notre carte de relais. Il faudra voir si la carte est théoriquement capable de supporter les intensités que nous lui faisons traverser.

J'ai également retravaillé les codes d'analyse d'images. Ils sont toujours découpés en trois parties, mais maintenant on peut spécifier plusieurs lignes à `1-TL.py` pour éviter d'avoir à relancer le process des images trop de fois. Par ailleurs, j'utilise maintenant numpy.savetxt pour sauver la trajectoire en csv dans `2-Traj.py`, ainsi que numpy.loadtxt pour la charger dans `3-WL.py`. J'ai aussi nettoyé le code de tous les commentaires de débugage inutiles.

Protocoles

Creation du film à partir des images

1. On utilise Thunar pour renommer nos fichier pour que leurs noms soient une suite numérotée ininterrompue (eg. 001.jpg, 002.jpg, etc)
2. On utilise la commande `ffmpeg -framerate 40 -i %03d.jpg -c:v libx264 -profile:v high -crf 20 -pix_fmt yuv420p output.mp4`

Les fichiers à utiliser sont données par l'option `-i`, on peut changer le framerate (ici 40 images par secondes).

Analyse des images de la nutation

1. On a mis toutes les photos du film dans un dossier. On récupère le chemin d'accès à ce dossier.
2. On utilise la commande : `python 1-TL.py Path Lines` avec `Path` est la localisation du dossier `Line` est une suite de lignes de pixel à extraire des photos (e.g. 500,600,700 sans espace, séparés par des virgules)
3. Puis `python 2-Traj.py Path Picture dt` avec `Path` la localisation du dossier, `Picture` est le nom de l'image générée par `1-TL.py`, et `dt` est le temps entre chaque photo.
4. Enfin `python 3-WL.py Path CSV` avec `Path` la localisation du dossier, `CSV` le nom du fichier csv créé par `2-Trajec.py`