|  |  |
| --- | --- |
|  | PEST Control |
|  |  |
| 22/01/2016 | 15dlm-TA-223 / Quentin Jeanmonod |
|  |  |

PEST Control

15dlm-TA-223 / Quentin Jeanmonod

# Résumé

Les ravageurs (*pest* en anglais) sont les insectes nuisibles à l'agriculture et notamment à la production de fruits (raisin, cerise, ...).

Si la population de certains peut être contrôlée grâce aux diffuseurs et pièges à phéromones, pour d'autres, comme la mouche Suzuki, il n'existe à l'heure actuelle pas de moyen efficace et écologique (pesticide à tout va) de limiter leur population.

PEST Control vise à intégrer dans un piège, un système de reconnaissance d'insecte qui permette d'estimer la population de mouches Suzuki en continu.

# Table des matières

Table des matières

[Résumé 1](#_Toc441182179)

[Table des matières 2](#_Toc441182180)

[1. introduction 3](#_Toc441182181)

[2. Analyse du problème et solutions étudiées 3](#_Toc441182182)

[2.1 Détection des insectes 3](#_Toc441182183)

[2.1.1 Détection à l’aide d’une barrière lumineuse 3](#_Toc441182184)

[2.1.2 Détection à l’aide d’une acquisition continue 4](#_Toc441182185)

[2.1.3 Solution choisie 4](#_Toc441182186)

[2.2 Performances 4](#_Toc441182187)

[2.2.1 Raspistill 4](#_Toc441182188)

[2.2.2 Utilisation de drivers open-source 5](#_Toc441182189)

[2.2.3 Détection de mouvement entre les images 5](#_Toc441182190)

[2.3 Qualité de l’image 6](#_Toc441182191)

[2.3.1 Résolution de l’image 6](#_Toc441182192)

[2.3.2 Éclairage 6](#_Toc441182193)

# introduction

Les ravageurs sont des animaux qui attaquent les plantes de culture ou les récoltes. La Drosophilia Suzukii est un ravageur qui s’attaque principalement aux fruits mûrissant, contrairement à la plupart des autres espèces de drosophiles qui s’attaquent aux fruits pourrissants. Cette espèce de mouche devient de plus en plus présente en Europe, et cause des dégâts importants aux cultures de fruits, notamment de raisins.

Afin de protéger les vignes et autres cultures de fruits mous, comme la cerise, la pêche ou l’abricot, il devient crucial de trouver une manière de contrôler la quantité de drosophile suzuki. Malheureusement, il n’existe pas à ce jour de phéromone attirant sélectivement ce ravageur pour contrôler les dégâts qu’il peut causer. À l’heure actuelle, on pose des pièges attirant tous types d’insectes, et des gens sont embauchés pour compter et différencier les insectes, afin de faire des statistiques sur les populations d’insectes. Cette approche a le problème d’être extrêmement coûteuse en temps et n’est effectuée qu’une seule fois par semaine.

L’idée derrière ce projet est de développer un piège permettant de compter les populations de drosophile suzuki automatiquement. Deux parties aux projets existent actuellement, une partie de traitement d’image qui doit différencier une drosophile suzuki d’une autre espèce d’insectes, et la partie dont traite ce présent rapport ; la détection de mouvements dans le piège pour capturer des images.

# Analyse du problème et solutions étudiées

Le piège prévu afin de détecter des insectes consiste à faire passer des insectes dans un tube, attirés par une substance attirante, de les photographier lorsqu’ils passent sous une caméra et d’envoyer les images à un serveur distant. Les points chauds de ce projet sont donc :

1. Détecter un insecte lorsqu’il passe au bon endroit ;
2. S’assurer d’avoir des performances suffisantes pour ne pas rater le passage d’un insecte ;
3. Avoir des images de qualités suffisantes pour reconnaître une espèce d’insecte.

Afin de facilement concevoir un système permettant de répondre à ce problème, on m’a confié un Raspberry PI 2, qui est un ordinateur de la taille d’une carte de crédit, prévu pour le développement embarqué. On m’a également confié une PiCam, une caméra plug’n’play spécialement prévue pour le Raspberry PI.

## Détection des insectes

Afin de détecter du mouvement, deux choix étaient possibles, soit utiliser une barrière lumineuse, soit effectuer une acquisition d’images en continu avec la caméra. La drosophile suzuki ne mesure en moyenne que 2.5 millimètres de long.

### Détection à l’aide d’une barrière lumineuse

Une barrière lumineuse consiste en 2 parties qui se font face, un émetteur lumineux et un capteur lumineux. Lorsqu’un objet passe entre les deux, le capteur ne reçoit plus de lumière et envoie un signal physique au Raspberry PI. Cette solution serait peu coûteuse en ressources, mais pose le problème suivant : si on place le capteur directement en dessous de la caméra, le déclenchement doit être instantané, si on le place en amont de la caméra, le comportement erratique de la mouche compromettra probablement un bon nombre de clichés.

### Détection à l’aide d’une acquisition continue

Cette solution a l’avantage d’être plus simple au niveau du développement hardware, mais coûteuse au niveau des ressources car il est nécessaire de comparer les images entre elles pour observer des différences afin d’identifier un mouvement.

### Solution choisie

Nous avons, le superviseur et moi, discuté avec des gens de l’école spécialisés dans le bas niveau, afin de déterminer la faisabilité de la solution hardware. Il en est ressorti qu’il serait probablement trop difficile de capter une mouche de cette taille avec un capteur bon marché. Nous avons donc choisi l’acquisition continue pour détecter les insectes.

## Performances

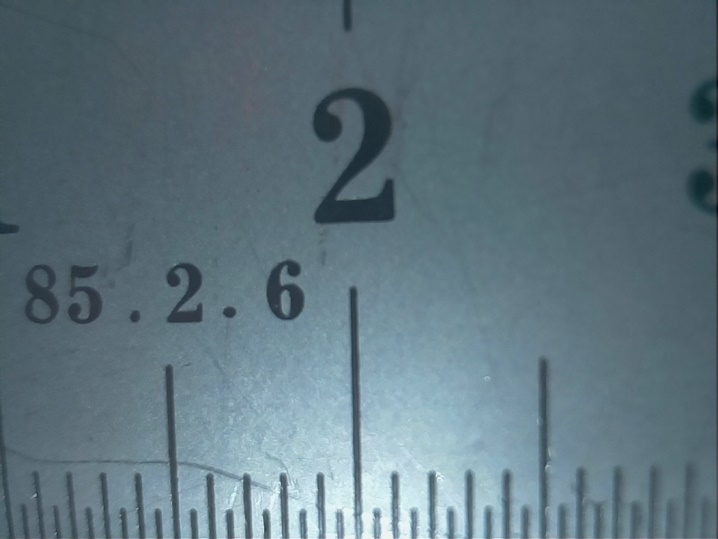
Puisque la solution choisie est de prendre des images en boucle, il est nécessaire d’avoir une certaine fréquence de capture afin d’être certain de ne pas rater un insecte qui passe en face de la caméra.

Figure 1: Largeur du champ visuel

Le champ visuel de la caméra fait approximativement 2 centimètres de large. La vitesse d’une mouche qui marche est d’environ 2 centimètres / seconde maximum[[1]](#footnote-1). Une mouche passant devant la caméra devrait donc prendre une seconde à traverser entièrement le champ visuel. La fréquence de capture requise est donc de 2 images / seconde minimum (théorème de Nyquist).

### Raspistill

Comme la PiCam est une caméra pour le Raspberry PI, il existe des commandes de base sur le PI permettant de l’utiliser. raspistill en est une. Elle permet de prendre une photo. Faire un programme utilisant cette commande serait la solution la plus simple pour capturer des images en continu, car il n’y aurait pas besoin de faire quoi que ce soit en bas niveau. Malheureusement, les premiers tests utilisant cette commande ont révélé qu’il y a un délai d’environ une seconde avant la capture. Cette solution n’est donc simplement pas possible pour effectuer une acquisition continue.

### Utilisation de drivers open-source

Après de longues recherches afin de trouver une solution plus rapide, j’ai trouvé des drivers s’utilisant avec OpenCV (Ce qui est parfait puisque cette bibliothèque permet de faire du traitement d’images, nécessaire pour détecter le mouvement entre les images). Le driver v4l2[[2]](#footnote-2) fut simple à installer et à utiliser, mais les premiers tests pas concluants. En effet, je n’arrivais qu’à un framerate d’environ 2 images / seconde. En descendant la résolution progressivement, il s’est avéré que le framerate saute à 30 images / seconde à 960x720 pixels. Cette solution semblait donc prometteuse au niveau de performances, et est utilisée dans la version finale du projet.

### Détection de mouvement entre les images

Afin de déterminer si mouvement il y a eu, il est nécessaire de faire une comparaison. Le procédé est le suivant :

* Prise d’une première image pour le fond de l’image, qui sert de référence.
* On prend le niveau de gris de cette image, et on effectue un flou gaussien dessus afin de réduire le bruit.
* Puis, pour chaque image, on effectue le même traitement, suivi de :
  + Une différence absolue des deux images (En gros, là où l’image est identique on a du noir, là où elle ne l’est pas, on a du blanc).
  + Seuillage de l’image.
  + Récupération des contours de l’image seuillée.
  + Calcul de l’aire englobée par ces contours. S’ils sont plus grands qu’une petite surface, ce n’est pas du bruit et on a détecté un mouvement.

Figure 2 : Niveau de gris flouté, différence et contours de de l'image seuillée

OpenCV fournit des méthodes pour chacune de ces étapes. L’opération la plus lourde en termes de ressources est le flou gaussien, qui est une convolution. Afin de garder une fréquence de capture suffisante tout en continuant de réduire le bruit, le noyau utilisé est relativement petit, c’est un carré de 5x5.

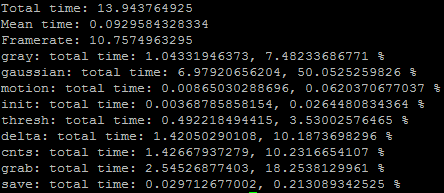
Un programme de test capturant 150 images donne les résultats suivants :

Figure 3 : Test de performances

Comme on peut le constater, le flou gaussien prend la moitié du temps, c’est-à-dire la moitié des ressources. Le framerate de presque 11 images / seconde nous assure qu’une mouche ne pourrait passer sous la caméra sans être détectée.

## Qualité de l’image

### Résolution de l’image

La qualité de l’image est primordiale pour effectuer un traitement d’image correct sur un insecte. Malheureusement, il n’était pas possible de pour des raisons de performances de capturer des images d’une résolution supérieure à 960x720. Les drivers utilisés sont pour l’instant les seuls disponibles pour utiliser la PiCam, et ils sont malheureusement encore très limités avec cette caméra. Je n’ai donc pu prendre nativement les images en niveau de gris, ce qui aurait peut-être permis d’augmenter la résolution, puisqu’une image en niveau de gris est 3 fois plus légère qu’une image couleur et que le problème de limitation vient peut-être du bus qui relie la caméra au PI. Cependant, mes recherches à ce sujet n’ont rien donné et ceci n’est que pure spéculation.

### Éclairage

Étant donné qu’on désire prendre beaucoup d’images par seconde et qu’un tube est un endroit dépourvu de lumières, il est nécessaire d’éclairer la scène. Une LED branchée au PI est donc une solution simple et économe. En effet, le PI disposant d’un GPIO[[3]](#footnote-3) et d’une bibliothèque python prévue pour, il n’est pas compliqué d’ajouter une LED au dispositif.

J’ai, dans un premier temps, utilisé une LED rouge que j’avais sous la main pour l’éclairage, puis une LED blanche dont la luminosité est plus grande et le spectre lumineux plus large. Comme on peut le constater sur les images suivantes, le fond est beaucoup plus bruité et dégrade sur le jaune sur le côté sur l’image où l’éclairage est rouge. Ceci est dû à la manière dont fonctionne le capteur optique. En effet, chaque pixel de l’image est encodée grâce à une matrice de Bayer[[4]](#footnote-4). Sur cette matrice, chaque pixel est encodé grâce à 4 capteurs uniquement sensibles à l’intensité de la lumière. Chacun de ces capteurs est derrière un filtre de couleur, un rouge, deux verts et un bleu. En éclairant avec une LED rouge, nous n’avons donc que le quart de l’information que si le sujet est éclairé avec une LED blanche.

Figure 4 : Test d'un point de stylo avec un éclairage rouge et d'une mouche avec un éclairage blanc

# Problèmes encourus

blablabla qt

1. <http://faculty.washington.edu/hueyrb/PDFAug2013/GibertEvol.pdf> (la Drosophilia Melanogaster est physiologiquement très semblable à la Drosophilia Suzukii) [↑](#footnote-ref-1)
2. https://www.raspberrypi.org/forums/viewtopic.php?t=62364 [↑](#footnote-ref-2)
3. https://www.raspberrypi.org/documentation/usage/gpio-plus-and-raspi2/ [↑](#footnote-ref-3)
4. https://fr.wikipedia.org/wiki/Matrice\_de\_Bayer [↑](#footnote-ref-4)