Système multimédia - TP6

January 7, 2019

Thomas COUCHOUD
thomas.couchoud@etu.univ-tours.fr
Victor COLEAU
victor.coleau@etu.univ-tours.fr

Chapter 1

Envoie d'une vidéo via le protocole UDP

1.1 Serveur

L'objectif de cette question est de réaliser un programme python permettant de filmer et de streamer en live sur 3 adresses différentes.

On commence par initialiser un socket avec les paramètres AF_INET (utiliser Internet) et SOCK_DGRAM (protocole UDP). Ensuite on créé un objet PiCamera est initialisons les paramètres suivants :

- **vflip** : inversion verticale de l'image.
- **hflip** : inversion horizontale de l'image.
- **resolution** : résolution de l'image. Ici à 640x480 pixels.
- **framefate** : framerate théorique de capture. Ici à 32 images par seconde.

Un objet rawCapture permettra par la suite d'obtenir tous les pixels de l'image enregistrée.

Une boucle For itérant sur la méthode camera.capture_continuous() permet de traiter chaque image tant que le flux vidéo est actif.

Pour chaque image, on commence par la passer en nuance de gris grâce à la méthode cv2.cvtColor() puis on la redimensionne grâce à la méthode cv2.resize().

Enfin, grâce à la méthode sock.sendto() appelée 3 fois, nous pouvons envoyer sur les 3 streams l'image précédemment traitée. Dans notre cas, les sockets transfèrent les données vers notre client, soit la VM utilisée, sur les ports 5006, 5007 et 5008.

1.2 Client

Pour le client, l'objectif est d'exposer un des 3 ports précédents afin de recevoir et afficher les données du stream.

On passe en paramètre le port souhaité et le stockons dans une variable.

On ouvre un socket similaire à celui du serveur et l'assignons à l'ip et au port voulus.

Ensuite, tant que le flux vidéo continu, on :

- Reçoit des octets grâce à la méthode sock.recvfrom(). La taille lue correspond à la taille d'une image.
- On créé une image à partir de ces données grâce à la méthode np.frombuffer().
- On redimensionne l'image en 320x180 pixels avec la méthode np.reshape().
- Enfin, on affiche l'image obtenu grâce à la méthode cv2.imshow().

Par soucis de commodité, nous avons ajouté une condition d'arrêt si l'on appuie sur la touche "q".

1.3 Serveur 2

Dans cette partie, plusieurs améliorations ont été ajoutées au serveur.

- Chaque stream est envoyé sur un port différent. Ici les ports 5006, 5007 et 5008. Pour ce faire, chaque stream est associé à un thread grâce à une classe *videoprocess* dérivée de *thread*.
- Le nombre d'images par seconde peut être choisit au lancement (premier paramètre du constructeur de l'objet *videoprocess*.

- Pour afficher un texte sur l'image, la méthode setImage de l'objet *videoprocess* appelle cv2.putText().

Après tests, pour 1 client, nous arrivons à environ 13 images par seconde.

Cependant, si l'on lance 3 clients simultanément, nous pouvons atteindre jusqu'à 10 images par seconde sur chacun d'entre eux.

1.4 Serveur 3

Dans cette version nous avons implémenté plusieurs changements:

- Max FPS de la caméra à 60
- Capture de la caméra directement à la résolution finale, permettant ainsi d'éviter les redimentionnements.

1.5 Screenshot

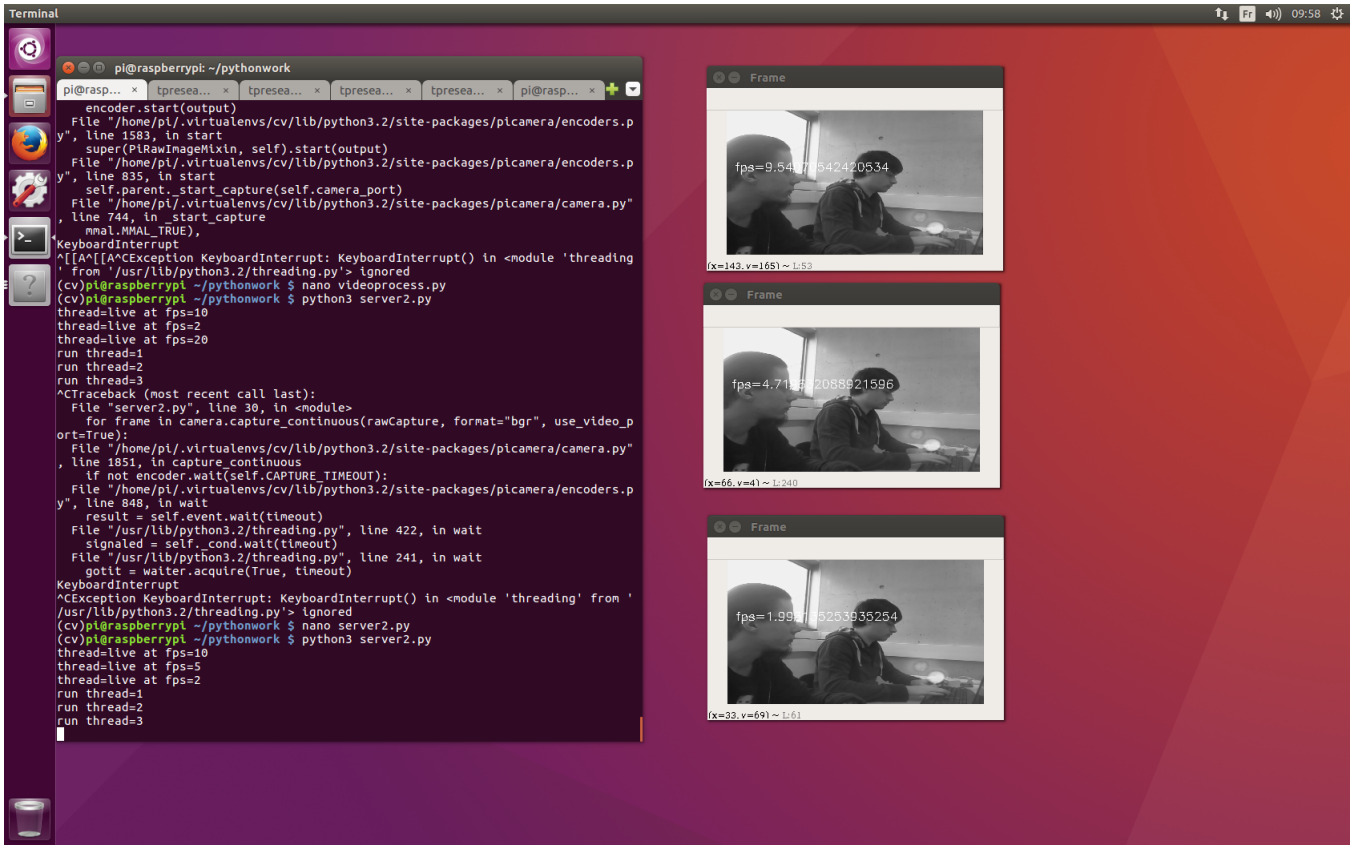

Figure 1.1