Détection de mouvement et méthode de Viola et Jones

# Résumé

Ce module capteur a pour but, à l’aide d’une caméra, de détecter le mouvement sur une portion de voie, et d’en déterminer la source, à l’aide de la méthode de détection d’objet de Viola et Jones. Pour ce faire, nous avons utilisé une plateforme Raspberry Pi, munie de sa caméra dédiée. Sur cette plateforme, nous avons implanté un programme utilisant une méthode de détection du mouvement simple (soustraction entre deux images), et un classifieur permettant de reconnaître le logo de l’ENSTA Bretagne. Ainsi, quand le logo est détecté (celui fixé à l’avant du train), la détection de mouvement suspect est désactivée tant qu’un deuxième logo (celui fixé à l’arrière du train) n’a pas été détecté. En revanche, quand un mouvement est détecté sans la présence du logo, un signal de mouvement suspect est émis à l’intention du serveur central.

Pour mettre en œuvre notre module, quatre parties principales ont été développées :

* L’algorithme de détection du mouvement et de prise en compte de la détection du logo
* Le classifieur permettant de détecter le logo
* L’interface entre le logiciel et la caméra du Raspberry Pi
* La partie réseau, permettant de relier le capteur au serveur central (pas encore implémenté)

Dans les parties suivantes, nous présentons l’implémentation de ce module.

# La plateforme Raspberry Pi

Le Raspberry Pi est un nano-ordinateur monocarte à processeur ARM conçu par le créateur de jeux vidéo David Braben, dans le cadre de sa fondation Raspberry Pi. Cet ordinateur, qui a la taille d'une carte de crédit, permet l'exécution de plusieurs variantes du système d'exploitation libre GNU/Linux et des logiciels compatibles. Il est fourni nu (carte mère seule, sans boîtier, alimentation, clavier, souris ni écran) dans l'objectif de diminuer les coûts et de permettre l'utilisation de matériel de récupération.

Dans notre implémentation nous avons choisi d’installer la distribution Linux créée spécialement pour la Raspberry Pi, Raspbian. Cette distribution présente l’avantage d’être facile d’utilisation et de pouvoir manipuler aisément l’ensemble des périphériques offerts sur la plateforme.

Le Raspberry dispose également d’une caméra qui lui est dédiée. Cette caméra s’adapte sur le port CSI de la carte.

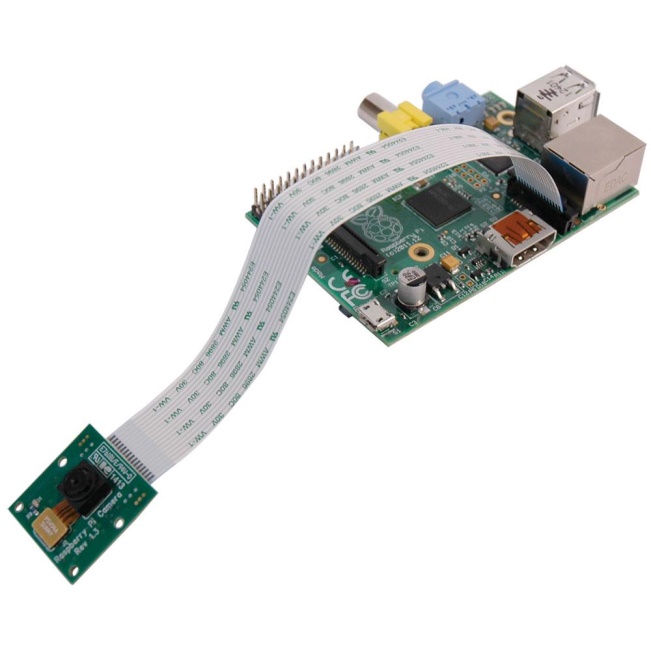


Figure 1 La carte Raspberry Pi, munie de sa caméra

La mémoire ROM de la Raspberry se trouve dans une carte SD non fournie. C’est sur cette carte SD que s’installent le système d’exploitation, ainsi que l’ensemble des fichiers et programmes.

Pour pouvoir manipuler les fichiers sur la carte, il existe plusieurs solutions.

* Ajouter manuellement les fichiers sur la carte SD (solution fastidieuse)
* Connecter l’ensemble des périphériques (clavier, souris, écran) au Raspberry, pour s’en servir comme d’un ordinateur normal.
* Se connecter en SSH au Raspberry, pour pouvoir le manipuler via son ordinateur. Il est également possible de manipuler facilement les fichiers grâce à un logiciel graphique comme FileZilla. Enfin, il est possible d’avoir le bureau graphique du Raspberry déporté. Pour cela, j’ai utilisé un serveur VNC. Le serveur TightVNC sur le Raspberry et VNCViewer sur l’ordinateur. Tous ces logiciels sont facilement disponibles dans les dépôts apt-get.

# L’interface Logiciel/Caméra

La plateforme Raspberry Pi étant assez récente, il n’existe pas encore de pilote officiel permettant de manipuler directement les images issues de la caméra. Aussi, la méthode utilisée pour manipuler les images en direct est généralement de streamer ces images, et de manipuler alors ce flux.

J’ai procédé d’une autre façon. La caméra est actuellement fournie uniquement avec un programme : camcv, qui permet de capturer des vidéos et des photos en ligne de commande. Un premier développeur, Pierre Raufast, a modifié ce programme, pour avoir un accès direct aux images capturées par la caméra. Puis, Cyril Rouvière, un élève de l’école, a simplifié ce code, dans le cadre d’un projet d’Odométrie sur robot. J’ai donc récupéré le code de son projet, pour ensuite l’adapter au mien. Ainsi, je peux avoir accès directement aux images capturées par la caméra, pour y appliquer les algorithmes de reconnaissance de logo et de détection du mouvement. C’est ce code qui m’a obligé à travailler en C++ sur ce capteur.

# L’algorithme de détection du mouvement

Le but de ce capteur est de pouvoir détecter le mouvement sur une portion de voie, mais également de pouvoir distinguer le train d’un élément extérieur. En effet, la détection de mouvement sur les voies autre que le train signifie un danger possible. Il est donc important d’en tenir compte dans la trajectoire du train, afin d’éviter les accidents. Enfin, il est important de pouvoir identifier le train. Pour cela, sur chacune de ses faces est collé un logo de l’ENSTA Bretagne qui devra être reconnu.

L’algorithme de détection du mouvement est modélisable par la figure suivante :

FL : F  
FM : F

FL : T  
FM : F

FL : F  
FM : T

Logo détecté

Logo détecté et non détection du logo entre-temps.

Logo détecté et détection du logo entre-temps.

Mouvement détecté

Logo détecté

X frames sans logo détecté  
ALERTE

Légende  
FL : Flag Logo  
FM Flag Mouvement

Logo détecté et détection du logo entre-temps.

Figure 2 Automate décrivant le fonctionnement de l'algorithme

C’est cet algorithme qui doit le plus prendre en compte l’environnement physique du train, et donc la dimension CPS. En effet, il est important de bien régler les délais en nombre de frames traitées avant l’émission d’une alerte. En effet, le mouvement du train sera toujours détecté avant que la caméra ne voit le train. De même, l’algorithme de détection du logo n’est pas infaillible. Il faut donc, avant de considérer qu’un nouveau logo a été détecté, attendre que l’ancien logo n’est pas été détecté pendant quelques frames consécutives, pour être sûr de ne pas avoir affaire au même logo. Ces paramètres doivent donc être réglés par l’expérience avec le système.

# Le classifieur

Pour entraîner notre classifieur, nous avons utilisé la bibliothèque C++ OpenCV, qui met déjà à disposition l’ensemble des outils pour entraîner un tel classifieur. Pour entraîner ce classifieur, nous avons tout d’abord utilisé 81 photos du logo ENSTA Bretagne, ainsi que 900 photos négatives ne contenant pas ce logo.

Au moyen du programme opencv\_createsample, nous avons alors créé 3000 images positives. Ces images positives sont obtenues en modifiant une photo du logo, et en l’insérant dans une image négative. Puis, nous avons entraîné notre classifieur, au moyen de l’outil opencv\_traincascade. A l’issue, nous obtenons un fichier .xml contenant notre classifieur. Il ne reste plus alors qu’à l’utiliser au moyen de l’algorithme de reconnaissance de Viola et Jones.

Pour notre premier classifieur, nous avons entraîné seulement 10 cascades, avec un taux de reconnaissance de 0,9 et un taux de faux positif de 0,5. Il est donc encore possible d’améliorer grandement ce classifieur.

# La partie réseau

Cette partie reste à implémenter. Je pense créer un client TCP/IP, qui se connecterait au même serveur Java précédemment développé. Ainsi, chaque client ne se connecterait au serveur que quand il a une alerte à émettre, et se déconnecterait dans la foulée. Ceci permettrait de limiter le trafic, et le nombre de connexions en parallèle, ce qui permettrait de multiplier le nombre de capteurs.