Détection de visages embarquée sur un Raspberry Pi

Pierre Ecormier, sous la direction de Dominique Béréziat
M1 - Projet d'Imagerie

25 mai 2016

Chapitre 1

Description du projet

1.1 Présentation

1.1.1 Objectif

L'objectif de ce projet est d'implémenter sur Raspberry Pi un algorithme de détection de visages humains sur un flux vidéo capturé via un module de caméra monté sur la tête d'un robot. Le projet sera utilisé en tant que demonstration et permettra à un robot chien de détecter puis de suivre du regard les visages des personnes se trouvant devant la caméra. La détection doit se faire sur des personnes se situant entre 1 et 2m de la caméra, et le framerate attendu est dans les alentours de 10 à 15 fps.

L'algorithme implémenté doit pouvoir fonctionner en temps réel sur le Raspberry Pi. Le choix des technologies est libre mais afin d'améliorer les performances du programme, l'utilisation des langages C ou C++ a été fortement conseillée.

1.1.2 Spécifications

Le robot est fourni par le Master de STL et est destiné à la coupe de France de robotique 2016 [1]. C'est un robot chien contrôlé par un Raspberry Pi et intégrant une caméra installée sur sa tête, qui peut tourner selon deux degrés de liberté.

La caméra est le module par défaut destiné au Raspberry Pi et possède un capteur de 5 mégapixels. Elle peut prendre des images statiques jusqu'à une résolution de 2592x1944 pixels, mais supporte également des résolutions inférieures pour permettre une capture en temps réel (1080p à 30fps, 720p à 60fps et 480p jusqu'à 90fps).

Les conditions de la démonstration sont considérées optimales, donc avec un éclairage favorable.

Les positions des visages détectés devront être passés à un programme externe qui s'occupera de la rotation de la tête du chien en fonction de la location des visages à l'écran. Le chien ne pouvant suivre du regard qu'une personne, on pourra donner en sortie une liste des visages trouvés ou un visage parmi ceux trouvés, selon un critère arbitraire (par exemple garder uniquement le plus grand et donc plus proche visage).

1. http ://www.planete-sciences.org/robot/index.php ?section=pages&pageid=133

1.1.3 Technologies utilisées

Le projet est développé en C++, avec l'aide de la librairie OpenCV. De plus, les fichiers de configuration des modèles de détection de visages fournis avec la librairie ont été réutilisés afin de ne pas avoir à refaire la phase d'apprentissage, une large base d'images annotées étant requise pour obtenir des résultats satisfaisants.

Des versions intermédiaires du projet ont été prototypées en Python et sans se soucier des contraintes liées au Raspberry Pi afin d'accélérer le développement et permettre de tester différentes approches plus facilement.

Bien que le projet final était destiné à fonctionner sur les derniers Raspberry Pi 3, le développement a dû être effectué sur un Raspberry Pi 1 à cause de délais de livraison des nouveaux modèles. En revanche, un module de caméra similaire à celui utilisé pour la démonstration a pu être fourni par le laboratoire STL pour permettre des tests en conditions quasi-réelles, et l'adaptation au materiel propre au Pi.

Chapitre 2

Algorithmes de détection de visages

2.1 Détection par couleur

La détection par couleur utilise plus d'informations qu'une détection « classique » pour la détection et peut donc être plus rapide selon l'implémentation (par exemple en réduisant les zones de recherche aux régions contenant des couleurs de peau).

Les méthodes se basant uniquement sur la couleur (ne s'en servant pas uniquement pour accélérer la recherche avec d'autres méthodes) peuvent également permettre d'être invariant par rotation.

Néanmoins ces méthodes peuvent être dangereuses car une couleur de peau non prévue ou un éclairage particulier peuvent faire échouer l'algorithme très facilement. Beaucoup de faux positifs peuvent également apparaitre selon l'arrière plan de l'image.

Un example d'algorithme basé sur la couleur de peau, proposé par Jay P. Kapur [1], est le suivant :

- Différence entre l'image filtrée par un filtre médian et l'image originale
- Utilisation d'un filtre de couleur peau via les valeurs HSV
- Extraction des régions de peau avec des thresholds
- Utilisation des « trous » dans les régions de peau (correspondant aux sourcils, bouche, etc) pour différencier les visages d'autres régions

Un autre exemple de détection de visage par couleur basé sur l'ACP a été proposé par B. Menser et F. Muller [2].

2.2 Détection via un modèle

Les méthodes de détection de visages par modèles semblent moins utilisées que leurs alternatives. Un exemple d'une telle méthode est décrit dans *Robust Face Detection Using the Hausdorff Distance* [3]. Cet algorithme détecte les contours dans une image, puis essaie d'appliquer des modèles de visages sur l'image obtenue en se basant sur la distance de Hausdorff

1. *Face detection in color images*, Jay P.Kapur
2. *Face Detection In Color Images Using Principal Components Analysis*, B. Menser and F. Müller
3. *Robust Face Detection Using the Hausdorff Distance*, Oliver Jesorsky, Klaus J. Kirchberg, and Robert W. Frischholz

(une métrique permettant de mesurer l'éloignement de deux sets dans l'espace) afin de trouver les similarités entre les sets de points.

Une méthode se basant sur les descripteurs SIFT [4] a également été développée, mais est surtout utilisée pour de la classification de visage plus que pour la détection.

2.3 Détection via une suite de classifieurs simples

La méthode présentée par Paul Viola et Michael J. Jones en 2003 [5] semble être la méthode la plus efficace pour de la détection en temps réel de visages. Elle est utilisée pour la détection de visages sur les appareils photos, et est l'implémentation utilisé par défaut dans OpenCV et Matlab.

L'utilisation de cet algorithme permet de réutiliser les classifieurs entrainés fournis par OpenCV (sous la forme de fichiers XML) et ainsi éviter la phase d'entrainement, qui nécessite une grande base d'images pour avoir de bons résultats.

Cette méthode est découpée selon les étapes suivantes :

- Utilisation d'une base de donnée de visages, et d'une base de donnée d'autres images qui sera utilisée comme base d'exemples négatifs
- Calcul de « l'image intégrale », qui permettra de calculer la somme des pixels de n'importe quelle région de l'image en temps constant
- Utilisation d'un algorithme de boosting, AdaBoost pour créer un classifieur puissant à partir de classifieurs simples. Pour cet algorithme, 3 types de classifieurs faibles (classifieurs constitués de 2, 3 ou 4 rectangles blancs et noirs) sont passés sur toute l'image à différentes tailles.
- Le mode de recherche de visage en cascade permet de rapidement exclure les régions ne contenant pas de visage et passer plus de temps sur les régions intéressantes

Le principal inconvénient de cette méthode est sa sensibilité à la rotation. L'algorithme peut détecter des visages inclinés jusqu'à un peu moins de 45°, mais n'est pas complètement invariant à la rotation. Aucune recherche ne semble avoir résolu ce problème tout en conservant la rapidité de l'algorithme, mais il est possible de le relancer sur des images inclinées à différentes rotations afin de détecter toutes les orientations possibles.

Il est également possible d'accélérer la recherche en cherchant en priorité les visages dans les zones ou d'autres visages ont été détectés dans les frames précédentes. Alternativement, une détection plus sensible mais invariante par rotation peut être effectuée en utilisant un algorithme de tracking dans l'image une fois que des visages ont été trouvés.

4. *Face Recognition using SIFT Features*, Mohamed Aly
5. *Robust Real-Time Face Detection*, Paul Viola et Michael J. Jones, 2003

Chapitre 3

Implémentation du projet

3.1 Spécificités liées au Raspberry Pi

Une fois les premiers prototypes effectués sur un ordinateur portables, le port de ce code sur un Raspberry Pi pour faire des essais en conditions réelles a posé plus de problèmes que prévu. En effet, le module de caméra fourni pour le projet est connecté à l'appareil via une nappe et non par usb comme l'attend OpenCV. La librairie est donc incapable de reconnaitre le périphérique et de détecter le flux vidéo.

Pour contourner le problème, nous avons repris comme base le code source des logiciels fournis avec le Raspberry Pi qui interagissent avec le module de caméra (*raspistill* et *raspivid*). À partir des sources de ces logiciels, nous avons donc pu intégrer notre propre code et passer à OpenCV les images dans le format attendu, permettant ainsi une capture à des taux de rafraichissement optimaux.

Le code initialement écrit pour un ordinateur portable a ensuite dû être adapté et optimisé afin d'avoir des performances acceptables sur le Raspberry Pi. La première implementation testée tournait à 4 images par seconde, mais 17 images par seconde ont pu être atteints après quelques modifications (principalement de la taille de fenêtres de recherche). Afin d'obtenir ces performances, les images capturées par le Raspberry Pi ont dû être réduits à une résolution très faible (320x240) et convertis en nuances de gris. Heureusement, la détection de visages semble fonctionner correctement malgré la faible résolution, et l'image doit être convertie en niveaux de gris de toutes façons pour être traitée par l'algorithme de Viola et Jones.

3.2 Détection de visages par la méthode Viola-Jones

Plusieurs tests ont été réalisés en Python et en C++ pour effectuer rapidement des mesures sur les limites et performances de l'algorithme. Les méthodes standard fournies par OpenCV, utilisant l'algorithme décrit par Viola et Jones, ont ainsi été testées avec différents paramètres afin d'optimiser les résultats et améliorer les performances.

De nouvelles techniques ont également été implémentées afin de résoudre un des principaux problèmes de cette méthode : la sensibilité à la rotation. Tout d'abord, l'algorithme a été testé afin d'étudier ses limites (environ 30 degrés avant que les visages ne soient plus reconnus, dans de bonnes conditions) en terme de résistance à la rotation. L'algorithme a ensuite été ap-

pliqué a différentes versions de l'image originale, qui ont été préalablement tournées sur elles mêmes afin d'annuler la rotation des visages. Les visages détectés sont ensuites re-transformés dans le système de coordonnées d'origine afin d'avoir, dans l'image de base, une détection de visages même s'ils ont subi une rotation. Deux angles de rotation à $\pm 22^\circ$ fournissent une robustesse suffisante tout en ne consommant pas trop de processeur. Pour plus de robustesse on peut également choisir une reconnaissance à trois rotations, à -45°, 0° et 45° mais à ce point les temps de calculs deviennent importants.

Sur le Raspberry Pi, la correction de la sensibilité à la rotation demande des calculs trop importants, et a donc dû être supprimée. Changer les tailles minimales et maximales de la fenêtre de recherche permet de réduire les possibilités à explorer et ainsi alléger les calculs. Ces paramètres permettent également de borner les distances auxquelles les visages vont être détectés. Les autres paramètres de l'algorithme permettent de d'affiner la qualité de la recherche, là encore en choisissant entre performance et qualité.

Si cette méthode fournit une détection solide des visages, avec une résistance à des faibles rotations de la tête à détecter, il est très lourd en terme de calculs. Les paramètres permettent d'améliorer les performances, mais la perte de qualité est significative et trouver les paramètres donnant le meilleur rapport qualité/performance n'est pas facile. Pour tenter de contourner ce problème, une deuxième approche à été testée et implémentée, avec en tête l'idée de combler les défauts de la méthode Viola-Jones tout en gardant son pouvoir de détection.

3.3 Tracking des visages avec l'algorithme KLT

L'algorithme de tracking KLT (Kanade-Lucas-Tomasi feature tracker) permet de suivre des points d'intérêt dans les images consécutives d'une vidéo. En combinant cet algorithme avec la détection de visages de Viola-Jones, il est possible de diminuer l'utilisation de l'algorithme de Viola-Jones, qui est très gourmand en calculs, et de seulement tracker les points des visages trouvés afin d'accélérer l'exécution. Cette méthode permet de garder la qualité de la détection initiale de Viola-Jones, puis de l'approximer pour les images suivantes.

Ce nouvel algorithme permet en effet d'augmenter les performances du programme, mais si la détection par Viola-Jones n'est pas renouvelée régulièrement, les points suivis s'écartent de plus en plus de leur position initiale et finissent par ne plus du tout être sur le visage à suivre (ou même complètement hors de l'image). Pour éviter que les points trop écartés du visage faussent le tracking, les points atteignant les bords de l'images sont considérés comme invalides et donc supprimés de la liste de points à suivre. De plus, si le nombre de points suivis devient trop faible, l'algorithme retourne à de la détection de visages basée sur Viola-Jones, jusqu'à qu'un visage stable soit trouvé pour relancer du suivi KLT sur ce visage.

Chapitre 4

Fonctionnalités additionnelles

4.1 Export des données

Afin d'interfacer notre application pour la démonstration finale, il est nécessaire d'exporter les positions des visages, qui seront ensuite récupérées par le programme s'occupant de la rotation de la tête du chien. Pour cela, il faut à chaque image, ou à un intervalle régulier, trouver le visage le plus gros et exporter ses coordonnées et positions. Ces informations peuvent être écrites dans un fichier qui sera monitoré par l'autre programme, transmises par socket pour éventuellement supporter ultérieurement une architecture client-serveur ou encore par des pipes (nommés ou juste des pipes shell en écrivant simplement sur la sortie standard).

La méthode de transmission de données choisie est l'écriture sur la sortie standard, pour que le programme puisse facilement être intégré dans des scripts shell. Le format choisi est le suivant : `x:01, y:23, w:45, h:67`, où `x` et `y` représentent les coordonnées du visage (le point en supérieur gauche du rectangle englobant) et `w` et `h` représentent respectivement la largeur et la hauteur du visage. Ce format à été choisi pour permettre d'être lisible par des humains, tout en restant facile à interpréter par un programme. En effet, la taille du texte entourant les données elle mêmes reste fixe et une simple conversion de chaine de caractère à entier permet de retrouver les valeurs voulues.

4.2 Interface de test via ligne de commande

Pour faciliter les tests et les démonstrations locales, une interface de lancement permettant de modifier les paramètres de la détection sans avoir à recompiler le programme permet un gain de temps important (en particulier sur le Raspberry Pi, peu puissant, ou le temps de compilation est bien plus grand que sur un ordinateur plus puissant). Une simple interface en ligne de commande semble être une bonne solution, permettant des changements rapides de paramètres, tout en ne demandant qu'un temps de développement relativement court. Une interface graphique ou des raccourcis clavier permettraient de modifier le programme en temps réel sans même avoir à le relancer, mais ces méthodes sont bien plus complexe à implémenter et n'entrent pas dans le cadre de ce projet.

Une interface en ligne de commande a finalement été implémentée, permettant de paramétrer l'algorithme de détection de Viola-Jones, et permettant d'activer ou non le tracking avec KLT. Le programme s'appelle de la manière suivante :

```
./camcv
  --minsize=xx
    # Taille minimale de la fenêtre de recherche, en pixels
  --maxsize=xx
    # Taille maximale de la fenêtre de recherche, en pixels
  --scalefactor=1.xx
    # Rapport d'agrandissement de la fenêtre de recherche
  --minneighbors=x
    # Nombre minimal de voisins nécessaires pour la détection
  --disabletracking=false
    # Booléen, détermine si le tracking KLT est utilisé
```

Conclusion

Ce projet de détection de visages embarquée sur un Raspberry Pi a soulevé de nombreux problèmes, dont notamment ceux liés à la détection même des visages. L'algorithme de Viola-Jones s'est trouvé être le plus adapté à ce problème particulier grâce à ses bonnes performance et sa très bonne détection. Cet algorithme rencontre néanmoins ses limites lors de l'utilisation sur un Raspberry Pi (en partculier les anciennes versions, moins puissantes). Afin de contourner ces défauts de performance, l'algorithme de tracking de Kanade-Lucas-Tomasi a été utilisé en complément pour essayer de réduire l'utilisation de Viola-Jones en approximant les positions des visages à certaines images.

Ce problème est lié à d'autres problèmes issus de l'utilisation d'un Raspberry Pi, qui a nécessité de nombreux ajustements. Le principal défi du Raspberry Pi était de faire fonctionner OpenCV avec le module de caméra fourni en gardant une bonne performance. Une solution à ce problème a pu être trouvée en utilisant comme base les logiciels fournis avec le Raspberry Pi, conçus pour exploiter le module de caméra avec de bonnes performances.

Au final, ce projet était une très bonne occasion d'en apprendre plus sur la détection de visages, d'objets en général (les idées utilisées pour ces techniques sont transposables à des détections d'objets plus génériques), ainsi que sur le développement embarqué et les problématiques liées au Raspberry Pi.