Compte rendu de l’avancée du programme d’analyse de vidéo

En reprenant la structure du code de l’an dernier, voici les remarques, modifications et difficultés que j’ai rencontrées.

Tout d’abord le code de l’an dernier était assez erroné, de par les paramètres et les méthodes de traitement choisies. Si le traitement des frames pour donner une binarisation (*mask*) fait appel à la bonne méthode, le problème vient du seuil de détection qui était bien trop élevé, tout du moins pour l’utilisation avec des caméras optiques.

Après on traite l’image binaire pour pouvoir reconnaître plus facilement les formes. Au début était mis en place de faire 50 érosions suivi de 50 dilatations, même en réduisant le nombre d’itérations, rien ne s’affiche en sortie, au mieux la binarisation est très peu modifiée, cela venait aussi du fait que l’élément de structure *kernel\_dilate* était assez petit. (*mask1*). Cela marche beacoup moins bien lorsqu’il y a peu de différences de couleurs (exemple avec le TEST1)

J’ai essayé plusieurs autres façons de traiter cette image binaire. Avec une fermeture suivie d’une ouverture, avec des éléments de structures assez grands, (des disques de rayon 50 pixels pour la fermeture et 20 pour l’ouverture) (c.f *mask2*) . Voici ce que l’on a en sortie avec des frames choisies pour leur difficulté de traitement (peu de différences avec l’image d’origine à cause de couleurs claires, plusieurs personnes).

*kernel\_dilate=np.ones((5, 5), np.uint8) #carré de coté 5*

*kernel\_morphcl=disk(75) #disque de rayon 75*

*kernel\_morphop=disk(15)*

*###########################################################################*

*#première méthode*

*mask1=cv2.erode(mask,kernel\_dilate, iterations=30)*

*mask1=cv2.dilate(mask1, kernel\_dilate, iterations=30)*

*#deuxième méthode*

*mask2=cv2.morphologyEx(mask, cv2.MORPH\_OPEN, kernel\_morphop)*

*mask2=cv2.morphologyEx(mask2, cv2.MORPH\_CLOSE, kernel\_morphcl)*

TEST1 :

|  |  |
| --- | --- |
|  |  |
|  |  |

TEST2 :

|  |  |
| --- | --- |
|  |  |
|  |  |

TEST3 :

|  |  |
| --- | --- |
|  |  |
|  |  |

TEST4 :

|  |  |
| --- | --- |
|  |  |
|  |  |

Il est à noté que j’ai coupé les images pour qu’elles apparaissent dans ce document, mais il sera nécessaire de traiter les images en entier, pour ne pas couper les personnes.

On peut penser que le masque 2 est assez convenable, le problème étant par exemple que si l’on veut bien différencier deux personnes (sur le TEST3), on doit réaliser une ouverture avec un élément structurant plus grand, le problème étant qu’avec cela, on risque de ne pas détecter les personnes sur les frames avec peu de différence de couleur, comme c’est le cas sur le TEST1), si l’on réalisait une ouverture avec un SE trop grand, on aurait juste un  ‘’anneau’’   affiché.

Ensuite l’algorithme va essayer de détecter les contours et créer un rectangle si les surfaces détectées sont assez grandes. Je n’ai pas encore bien vérifié ou modifié cette partie, mais déjà en voyant les premiers résultats, on peut tout d’abord imposer une limite sur le rapport entre la longueur et la largeur de ce rectangle. Selon l’orientation de la caméra, les cas difficiles où il y a par exemple deux personnes va résulter avec l’algorithme actuel sur le TEST2 en la création d’un grand rectangle horizontal, comptabilisant de ce fait une personne au lieu de deux.

Il faudra aussi prendre en compte la puissance de calcul sur les cartes, ici j’ai traité les vidéos avec mon pc, ce qui s’est fait plus ou moins en à vitesse réelle si j’affichais les images différentes étapes de transformations en temps réel. Il faudra donc sans doute diminuer le nombre d’images analysées, et diminuer la résolution. Il faudra donc bien veiller à modifier les paramètres de tailles comme celui des tailles des éléments structurants, ou bien des tailles des surfaces à éliminer.

D’après les recherches que j’ai pu faire en ligne, c’est une des seules méthodes valables pour la Raspberry pi, les algorithmes de détections se basant sur du machine learning sont plus performants mais totalement inadaptés à notre utilisation. Nous allons essayer de prendre d’autres vidéos pour voir si d’autres points de vue fonctionneraient mieux.

Il est à noter que cette méthode fonctionne bien pour compter le nombre de personnes qu’il y a dans une pièce, puisque le traitement ne prend pas en compte les frames qui se suivent mais juste la différence avec départ. Il faut aussi prendre en compte que le fond risque de changer en fonction de la luminosité, il faudrait donc actualiser le fond par défaut à minima toutes les 10 min lorsque la caméra ne captera pas de personnes.