Vidéo surveillance élémentaire

Introduction

Comme illustré par la figure 1, l'objectif est de détecter une intrusion (affichage "Intrusion") et de la localiser (cercle superposé à l'image affichée).

Afin de tester l'algorithme, on utilisera la video fournie. Un programme exemple dédié à la lecture et l'affichage d'une video est également fourni.

L'algorithme proposé comprend deux étapes (sections 1 et 2). La première étape concerne la détection d'intrusion, basé sur la calcul du mouvement, la seconde étape étant consacrée à la localisation de cette intrusion.

Pour tester l'algorithme en évitant de charger à chaque fois le flux vidéo complet, pensez à travailler sur des "snapshots" (le caneva fourni permet de générer des images "snapshots" en pressant la touche "a" lors de la lecture du flux vidéo).

English:
As illustrated by Figure 1, the purpose is to detect an intrusion (text "intrusion") and to localize if (circle superimposed on the displayed image). To test your work, the provided video can be used. An example program allowing to read and display an video is also provided. The proposed algorithm consists in two steps (sections 1 et 2). The first step is dedicated to the detection of the intrusion (based on movement detection), while the second step aims at localizing the intrusion. Instead of directly testing your algorithm on the video file, think about working on snapshots (the provided programs allows to get snapshots by pressing the "a" key)

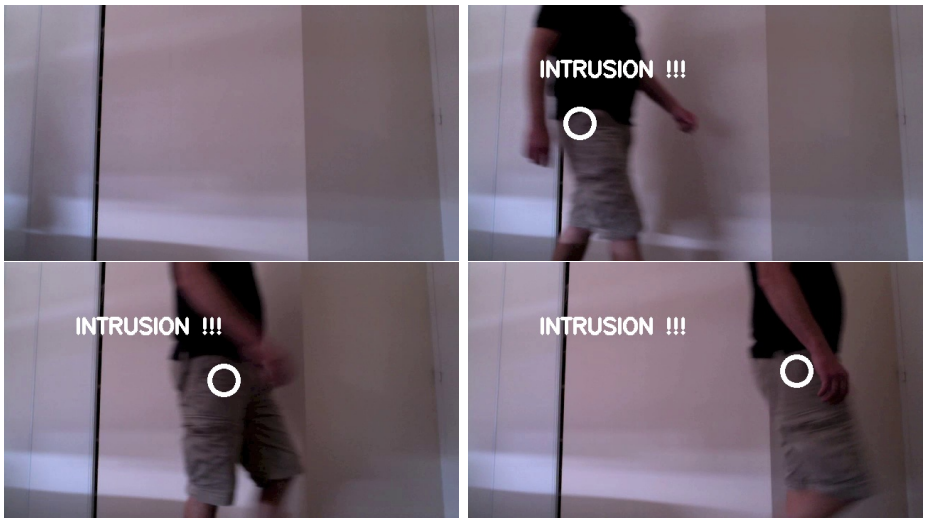

Figure 1: Détection et localisation d'une intrusion. Detection and localization of an intrusion.

1 Détection d'une intrusion - Detection of an intrusion

On commencera par calculer la variation d'intensité entre deux frames consécutives (voir figure 2):

$$\Delta_t(x, y) = |I_t(x, y) - I_{t-1}(x, y)|$$

Les images (frames) sont initialement codées sur 8 bits non signés ([0,255]). Il faut veiller à éviter les dépassements de dynamique lorsque l'on opère la différence entre les deux frames consécutives. Pour cela, il faut coder au préalable les images sur une plus grande dynamique supportant les valeurs négatives. Dans notre cas, il est suffisant de passer en entiers signés sur 16 bits (voir caneva fourni): `image=image.astype(numpy.int16)`.

On appliquera ensuite un seuillage (fonction `numpy.where`), pour ne retenir que les points correspondant à un mouvement significatif (i.e. changement significatif d'intensité): $\Delta_{t,th}(x, y) = 1$ si $\Delta_t(x, y) > T$, sinon 0. On pourra considérer un seuil de 40 (i.e. $T = 40$).

Figure 2: Deux images consécutives et leur différence. Two consecutive images and their difference

Enfin, on comptabilisera le nombre de points associés à un mouvement significatif (somme des points à 1 dans l'image $\Delta_{t,th}(x, y)$, à l'aide de `numpy.sum`). On considérera qu'il y a intrusion si un nombre suffisant de points de l'image sont associés au mouvement. Dans notre cas, si 1 % des points de l'image sont à "1", on considère qu'il y a intrusion.

Pour l'application au flux video complet, on règlera les paramètres de manière à afficher "INTRUSION" au moment opportun.

English:
On starts computing intensity variations between two consecutive frames (see Figure 2):

$$\Delta_t(x, y) = |I_t(x, y) - I_{t-1}(x, y)|$$

Frames are initially unsigned 8 bits images (i.e. [0,255] range). One has to avoid dynamic overflow when computing the difference between unsigned 8 bits images. For this, frames must be first casted into a larger encoding range, supporting negative values (signed 16 bits images should be considered) : `image=image.astype(numpy.int16)`.

Threshold the difference image (using `numpy.where`) to only consider pixels corresponding to significant movement (i.e. large intensity variation) :

$\Delta_{t,th}(x,y) = 1$ if $\Delta_t(x,y) > T$, 0 otherwise. One can consider a threshold value of 40 (i.e. $T = 40$).

Finally, count the number of pixels associated to a significant movement : number of pixels with a value of 1 in $\Delta_{t,th}(x,y)$ (using `numpy.sum`). One consider that there is an intrusion if this number is large enough (in our case, if more than 1% of the pixels are concerned). Adapt the program to display "INTRUSION" when this condition is satisfied.

2 Localisation de l'intrusion - localization of the intrusion

L'intrusion sera localisée par le barycentre de l'image seuillée des différences $\Delta_{t,th}(x,y)$. On calculera ce barycentre en utiliserant la fonction `numpy.where`.

L'utilisation de `numpy.where` dans ce cas différe de l'usage fait préalablement pour le seuillage: `numpy.where(image!=0)` retourne les tableaux d'indices `indices_i` et `indices_j` (ou x, y) pour lesquelles `image` est différent de 0. Le barycentre (xc, yc) peut alors être déterminé par:

```
xc=numpy.mean(indices_i)
yc=numpy.mean(indices_j).
```

Pour l'affichage du cercle localisant l'intrusion, il faut avoir conscience du systême de coordonnées (0,x,y). Dans le cas de `numpy` (utilisé pour les calculs), x désigne l'indice de la ligne et y celui de la colonne. Dans le cas d'opencv (utilisé pour l'affichage), x désigne la colonne et y la ligne. Dans les deux cas, l'origine 0 est le coin supérieur gauche de l'image.

English:

The intrusion will be localized by the center of mass of thresholded difference image $\Delta_{t,th}(x,y)$, being calculated using `numpy.where`. The use of `numpy.where` is not the same as for thresholding : `numpy.where(image!=0)` returns two numpy array of the indices `indices_i` and `indices_j` (or x, y) for which `image` is not null. The center of mass (xc, yc) can therefore by determined by:

```
xc=numpy.mean(indices_i)
yc=numpy.mean(indices_j).
```

To display the circle at the center of mass, one must take care of the coordinate system (0,x,y) (difference between opencv and numpy). With `numpy` (used for computations), x corresponds to rows and y to columns.

With opencv (used for display), x corresponds to columns and y to rows. In both cases, the origin is the top-left corner of the image.