Reconnaissance d'objet par la technique du « template matching »

Préparation d'un projet C++ sous Visual Studio

Il s'agit ici de créer un projet « Win32 application console C++ » et de ne pas oublier de configurer les répertoires « include » et « bibliothèque » comme nous l'avons fait précédemment.

Ajoutez un fichier source dans lequel nous y mettons les entêtes usuels :

```
#include <opencv2/highgui/highgui.hpp>
#include <opencv2/imgproc/imgproc.hpp>
```

Étape à suivre

Les principales étapes dans la détection de forme plus complexe résident essentiellement en prenant une image canevas (template) et en comparant cette dernière avec l'image source de l'objet recherché.

Étape 1 : Chargement de l'image source et de l'image canevas

Étape 2 : Préparer le tableau qui recevra les résultat. Cette image a toujours une dimension donnée par :

NbColTemplate = ImageSource->cols – ImageTemplate->cols +1
NbRangTemplate = ImageSource->rows – ImageTemplate->rows + 1

Préparer l'image qui recevra les résultats :

ImageResultat.create(NbColTemplate, NbRangTemplate, CV_32FC1);

Étape 3 (optionnel) : Appliquer un traitement de « smoothing » à l'image. Peut-être également une étape de morphologie en amaigrissant le contour de 1 pixel de largeur.

Étape 4 : Chercher l'objet à l'aide du canevas (l'opération de « matching » proprement dite)

Étape 5 (optionnel) : Normaliser l'image (si on veut afficher les résultats)

Étape 6 : Trouver la coordonnée du point qui représente l'endroit le plus probable pour la présence de l'objet à chercher.

Appel des fonctions

Étape 1 : Chargement de l'image source et de l'image canevas

```
ImgSource = imread...

ImgTemplate = imread...
```

Étape 2 : Préparer le tableau qui recevra les résultat. Cette image a toujours une dimension donnée par :

ImageSource->cols – ImageTemplate->cols +1
ImageSource->rows – ImageTemplate->rows + 1

```
NbColTemplate = ImageSource->cols – ImageTemplate->cols +1
NbRangTemplate = ImageSource->rows – ImageTemplate->rows + 1

ImageResultat.create(NbColTemplate, NbRangTemplate, CV_32FC1);
```

Étape 3 (optionnel) : Appliquer un traitement de « smoothing » à l'image. Peut-être également une étape de morphologie en amaigrissant le contour de 1 pixel de largeur.

Voir les fonctions : erode, dilate, morphologyEx

Étape 4 : Chercher l'objet à l'aide du canevas (l'opération de « matching » proprement dite)

```
matchTemplate(imgsource, imgTemplate, imgResult, CV_TM_CCORR );
```

Attention : Il se peut que d'autres algorithmes vous donnent de meilleure résultat. Il faut essayer. Ces algorithmes de recherche sont spécifiés par une constante que l'on indique dans le dernier paramètre de la fonction cvMatchTemplate. Parmi les algorithmes de recherche, il y a :

- CV_TM_SQDIFF
- CV_TM_SQDIFF_NORMED
- CV_TM_CCORR
- CV_TM_CCORR_NORMED
- CV_TM_CCOEFF
- CV_TM_CCOEFF_NORMED

La documentation indique que les résultats sont intéressant avec CV_TM_CCOEFF_NORMED mais j'ai également eu des résultats intéressant avec CV_TM_CCORR.

Étape 5 (optionnel) : Normaliser l'image (si on veut afficher les résultats)

```
normalize(imgResultat, imgResultat, 0, 1, NORM_MINMAX, -1, Mat());          //Plus pour
l'affichage
```

Étape 6 : Récupérer les résultats

```
minMaxLoc(imgResultat, &MinVal, &MaxVal, &MinLoc, &MaxLoc, 0);
```

Les résultats obtenus sont disponibles dans la structure qui est passé en avant-dernier paramètre (MaxLoc). C'est une structure de type Point définie comme suit :

```
struct Point
{
        int  x;   // Valeur x du point correspondant au « match »
        int  y;   // Valeur y du point correcpondant au « match »
};
```

Les paramètres de cette fonction sont définis comme suit :

minMaxLoc (imgResult, &MinVal, &MaxVal, &MinLoc, &MaxLoc, 0);

imgResult : Image qui contient le résultat de cette opération. Cette image ne contient généralement pas beaucoup d'information utile pour l'affichage. Elle doit être définie comme suit :

ImageResultat.create(NbColTemplate, NbRangTemplate, CV_32FC1);

Une image en ton de gris dont chaque pixel est codé sur une valeur de type float.

MinVal : La valeur minimale où se situe le pire « match ». Une valeur entre 0 et 1.

MaxVal : La valeur maximale où se situe le meilleur « match ». Une valeur entre 0 et 1.

MinLoc : Structure Point qui contient le point du pire « match ».

MaxLoc : Structure Point qui contient le point du meilleur « match ».

Attention : Pour les méthodes SQDIFF et SQDIFF_NORMED, les meilleurs « match » sont représentés par les valeurs minimales. Toutes les autres méthodes contiennent les meilleurs « match » dans les valeurs maximales.