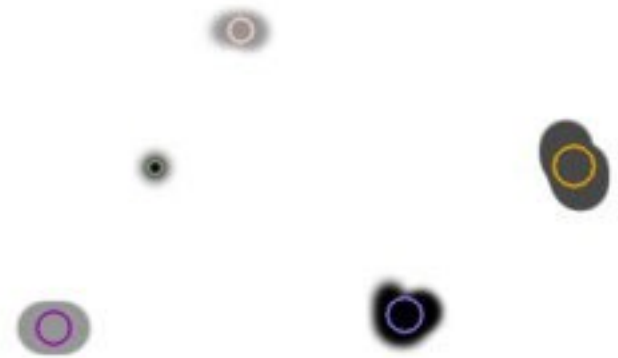
Raspberry Pi OpenCV Détection de blobs (taches)

1 Introduction à la détection de blobs



Qu'est-ce qu'un Blob? Un Blob est un groupe de pixels connectés dans une image qui partage une propriété commune par exemple la même valeur de gris. Dans l'image ci-dessus, les zones obscures sont des taches , L'objectif de la détection de blob est d'identifier et de marquer ces régions.

SimpleBlobDetector, comme son nom l'indique, est basé sur un algorithme assez simple qui est contrôlé par des paramètres (en gras ci-dessous) et comporte les étapes suivantes.

1. **Thresholding**: Les images source sont converties en plusieurs images binaires en seuillant l'image source avec des seuils commençant à **minThreshold**. Ces seuils sont incrémentés par **thresholdStep** jusqu'à **maxThreshold**. Donc, le premier seuil est minThreshold, le second est minThreshold + thresholdStep, le troisième est minThreshold + 2 x thresholdStep, et ainsi de suite.

2. Groupement: dans chaque image binaire, les pixels blancs connectés sont regroupés. Appelons les blocs binaires.

3. Fusion: Les centres des blocs binaires dans les images binaires sont calculés et les blocs situés à proximité de **MinDistBetweenBlobs** sont fusionnés.

4. Calcul du centre et du rayon: Les centres et les rayons des nouveaux blocs fusionnés sont calculés et retournés.

2 Filtrage des blocs par couleur, taille et forme

Les paramètres de **SimpleBlobDetector** peuvent être configurés pour filtrer le type de blobs que nous voulons.

Par couleurs : Cette fonctionnalité semble être cassée. J'ai vérifié le code, et il semble y avoir une erreur logique. D'abord, vous devez définir **filterByColor** = 1. Définir **blobColor** = 0 pour sélectionner des blobs sombres, et blobColor = 255 pour des blobs claires.

Par taille: Vous pouvez filtrer les blobs en fonction de la taille en définissant les paramètres **filterByArea** = 1, et les valeurs appropriées pour **minArea** et **maxArea**. Par exemple. Le réglage minArea = 100 filtrera tous les blobs qui ont moins de 100 pixels.

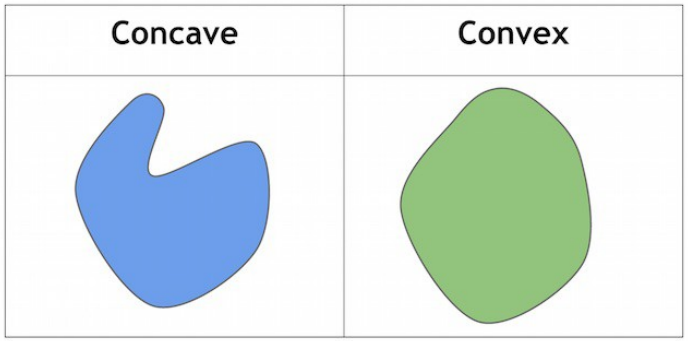
Par forme: la forme présente trois paramètres différents.

- **Circularité**: cela mesure à quel niveau un blob est un un cercle. Par exemple. Un hexagone régulier a une circularité supérieure à celle d'un carré. Pour filtrer par circularité, définissez **filterByCircularity** = 1. Ensuite, définissez les valeurs

Fiche 41 Raspberry OpenCV détection de taches

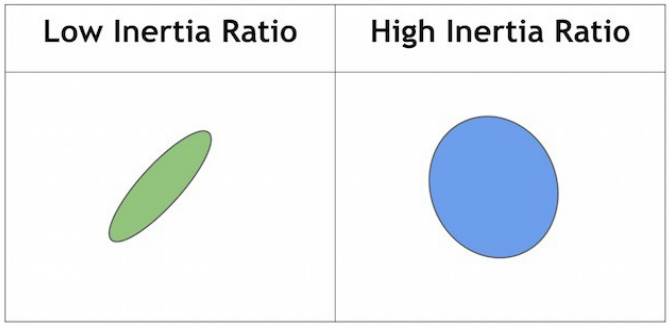
appropriées pour **minCircularity** et **maxCircularity**. La circularité est définie comme (\ frac {4 * \ pi * Zone} {périmètre * périmètre}). Cela signifie qu'un cercle a une circularité de 1, la circularité d'un carré est 0.785, et ainsi de suite.

- **Convexité**:



une image vaut mille explications. La convexité est définie comme le rapport entre (l'aire du Blob / l'aire de sa coque convexe). La coque convexe d'une forme est la forme convexe la plus étroite qui entoure complètement la forme. Pour filtrer par convexité, définissez **filterByConvexity** = 1, puis définissez 0 ≤ **minConvexity** ≤ 1 et **maxConvexity** (≤ 1).

- **Ratio d'inertie**:



Les mathématiciens utilisent souvent des mots confus pour décrire quelque chose de très simple. Tout ce que vous devez savoir, c'est que cela mesure l'allongement d'une forme. Par exemple. Pour un cercle, ce ratio vaut 1, pour une ellipse il vaut entre 0 et 1, et pour une ligne, il vaut 0. Pour filtrer par rapport à l'inertie, définissez **filterByInertia** = 1 et définissez 0 ≤ **minInertiaRatio** ≤ 1 et **maxInertiaRatio** (≤ 1) De manière appropriée.

3 Comment définir les paramètres de SimpleBlobDetector?

Dans OpenCV 3, la méthode SimpleBlobDetector :: create() permet de créer un pointeur intelligent Ptr

Un Ptr <T> est être un pointeur vers un objet du type T. Contrairement à un pointeur ordinaire, l'objet sera automatiquement nettoyé une fois que toutes les instances Ptr qui le pointeront seront détruites.

Programme exemple :

https://github.com/PhilippeSimier/openCV/tree/master/BlobDetector