

Projet d'analyse d'image

Détection et reconnaissance de panneaux de signalisation routière

CARLOS VALADARES , GABRIELLA BETTONTE

29 janvier 2019

1 Introduction

Ce projet s'inscrit dans le domaine d'étude du guide automatique : il vise à reconnaître automatiquement les panneaux de signalisation avec précision sans utiliser de techniques d'apprentissage automatique, en appliquant les connaissances théoriques acquises dans les cours de traitement d'images.

Diverses images contenant des panneaux de signalisation ont été fournies comme dataset. Les photos ont été prises à différentes distances, avec différentes illuminations et cameras pour simuler des situations pouvant se produire dans la réalité (regarder la figure 1). Il faut que notre application soit capable de détecter tous les signaux, qu'ils soient tournés, mal éclairés, lointains sans obtenir beaucoup de faux positifs, pour augmenter l'efficacité et le bon fonctionnement du système.

Pour le développement de l'application, on a utilisé Matlab.



FIGURE 1 – Pictures of roads

2 Procédure

2.1 Coupage

Après avoir observé les images, on a constaté que les panneaux de signalisation se trouvaient toujours dans la partie supérieure de l'image. On a donc décidé de couper les images à une hauteur égale au $\frac{3}{4}$ de la hauteur de l'image, fig.2.

Cette opération vise à réduire l'image à traiter, donc par conséquent le temps d'exécution, et on a semblé légitime car, puisque on est dans le domaine de la conduite automatique, il est plausible que les panneaux de signalisation à reconnaître, pendant que la voiture roule, se situent dans la partie supérieure de la vue et non à proximité du sol.

2.2 Normalisation

On a essayé de transférer l'image dans différents espaces, afin d'évaluer dans laquelle il était plus facile d'extraire les panneaux de signalisation rouges.

Au départ, on c'est concentré sur l'espace HSV, mais les résultats n'ont pas été satisfaisants pour de nombreuses images. On a également essayé de



FIGURE 2 – Coupage à une hauteur de $\frac{3}{4}$

normaliser l'image en divisant simplement les valeurs par 255, mais les résultats étaient mauvais.

Ensuite, on a transformé l'image RGB en une image RGB normalisée et on a obtenu de bons résultats.



FIGURE 3

FIGURE 4 – Normalisation

2.3 Seuillage

En suite, on a analysé l'image avec l'aide de le curseur matlab et on a choisi les bonnes seuils à la main :

- $RMeanThreshold = 0.63;$
- $GMeanThreshold = 0.55;$

— $BMeanThreshold = 0.6$.

On a sélectionné dans l'image les zones contenant une valeur rouge supérieure à $RMeanThreshold$ et les valeurs bleue et verte ci-dessous à $BMeanThreshold$ et $GMeanThreshold$ car on est intéressé à le panneaux rouge.

2.4 Filtrage

Ayant fait la seuillage, il faut traiter encore un peu l'image pour avoir un résultat plus précis. Pour cela, on applique la fonction `bwareaopen` pour enlever des petits objets, dans ce cas les objets connexes ayant moins de 200 pixels et ensuite, on applique un `erosion` avec un disque de taille 1.

Après, pour supprimer encore des petits objets on applique le filtre `Sobel`, pour prendre les bords des objets, suivi par une dilatation par un disque de taille 2 envisageant remplir les éventuels espaces entre les bords.

2.5 Post-traitement

Après avoir sélectionné les contours de l'image, comme expliqué au point précédent, on s'est consacré à la réalisation d'un post-traitement pour améliorer la détection des panneaux de signalisation.

Nous avons réalisé une dilatation avec un élément structurant de disque afin de fermer les contours pour délimiter parfaitement la zone souhaitée. Puis, on applique la fonction `imfill` pour remplir tous les trous (cette fonction ira remplir l'objet juste si son contour est tout connecté) et finalement on applique un `erosion` par un disque de taille 4 et un autre `bwareaopen` pour supprimer tous les objets qui n'ont pas les bords connectés, c'est-à-dire, tous les objets que n'ont pas été remplis par la fonction `imfill`.

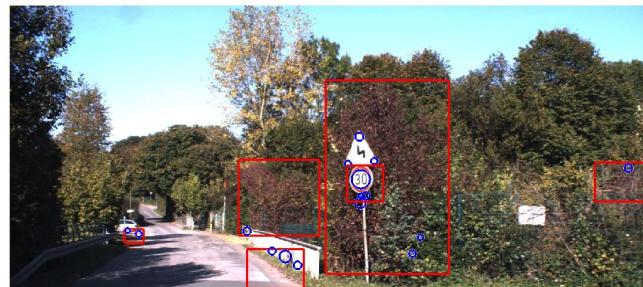
2.6 Bounding box

Envisageant trouver les bords de chaque région à analyser on utilise la fonction `regionprops` avec les propriétés **Area** et **BoundingBox**.

Pour diminuer le temps de calcul on filtre les régions rentrées par cette fonction, on n'ira pas analyser les régions qu'ont un aire de moins de 180 pixels et aussi si la région est très allongé sur l'horizontal ou vertical. Ensuite, on analyse chaque petite région en cherchant des panneaux.

Pour chaque enveloppe convexe qu'on a trouvé comme potentiel panneau on applique la fonction `imfindcircles` qui applique une *Hough Transform* sur

seulement cette partie de l'image. Si on trouve des cercles dans la région c'est probable qu'il y a une panneau. Avec cet approche on peut détecter les panneaux dans 90% des photos fournies comme base de données, cependant il y avait pas mal de faux positives dans certaines images, fig 5a. Donc, Envisageant l'élimination des faux positives on extract les caractéristiques SIFT des cercles et les compare avec une image base. Cela nous a permis d'être plus précis, mais a impliqué aussi en une baisse de la taux de détection. En supposant que la détection des faux positives est plus grave que la détection de la plaque on a gardé cet dernière approche. Comme résultat final on a réussi à détecter les panneaux rond et rouges dans 31% des images fournies comme base de données.



(a)



(b)

FIGURE 5 – Résultats

3 Liste de toutes les fonction MATLAB utilisées

3.1 RGBNormalize(rgbImage)

Cette fonction matlab *RGBNormalize*, qui on a écrit, a pour paramètre d'entrée l'image originale et elle renvoie l'image normalisée.

On a donc constaté que la normalisation donnant de bons résultats est la suivante : $\left(\frac{R}{\sqrt{R^2+G^2+B^2}}, \frac{G}{\sqrt{R^2+G^2+B^2}}, \frac{B}{\sqrt{R^2+G^2+B^2}} \right)$. On peut appeler cela une normalisation car la somme de ces trois fractions donne 1.

3.2 conv2(A, B) :

renvoie la convolution à deux dimensions des matrices **A** et **B**.

3.3 uint8 (X) :

convertit les valeurs dans X en uint8. Les valeurs situées en dehors de la plage [0,28-1] correspondent au point final le plus proche.

3.4 BW2 = bwareaopen (BW, P) :

supprime tous les composants connectés (objets) qui ont moins de P pixels de l'image binaire BW, produisant une autre image binaire, BW2. Cette opération est appelée ouverture de zone.

3.5 J = imerode (I, SE) :

érode les images en niveaux de gris, binaires ou binaires compacts I, renvoyant l'image érodée. J. SE est un objet d'élément structurant ou un tableau d'objets de structuration, renvoyé par les fonctions strel ou offsetstrel.

3.6 BW = edge(I) :

renvoie une image binaire BW contenant des 1 où la fonction trouve des bords dans l'image d'entrée I et des 0 ailleurs. Par défaut, edge utilise la méthode de détection de bord Sobel.

3.7 J = imdilate (I, SE) :

dilate les images en niveaux de gris, binaires ou binaires condensées I, renvoyant l'image dilatée, J. SE est un objet d'élément structurant ou un tableau d'objets de structuration, renvoyé par les fonctions strel ou offsetstrel. Ici on a utilisé comme élément structurant un cercle de rayon 2.

3.8 BW2 = imfill (BW, 'holes') :

remplit les trous de l'image binaire d'entrée BW. Dans cette syntaxe, un trou est un ensemble de pixels d'arrière-plan qui ne peut pas être atteint en remplissant l'arrière-plan à partir du bord de l'image.

3.9 J = imopen (I, SE) :

effectue une ouverture morphologique sur l'image en niveaux de gris ou binaire I, renvoyant l'image ouverte, J. SE est un objet élément structurant unique renvoyé par les fonctions strel ou offsetstrel. L'opération morphologique ouverte est une érosion suivie d'une dilatation, utilisant le même élément structurant pour les deux opérations. Ici on a utilisé un cercle de rayon 4 comme élément structurant.

3.10 stats = regionprops (BW, propriétés) :

renvoie les mesures de l'ensemble des propriétés spécifiées par les propriétés de chaque composant (objet) connecté à 8 dans l'image binaire, BW. Stats est un tableau struct contenant une structure pour chaque objet de l'image. Comme propriété on a pris 'BoundingBox' : le plus petit rectangle contenant la région, renvoyé sous forme de vecteur 1 par $Q * 2$, où Q est le nombre de dimensions de l'image. La fonction retourne [gauche, haut, largeur, hauteur] du rectangle.

3.11 imfindcircles(I, range) :

trouve les cercles dans l'image I qu'ont le rayon dedans l'intervalle passé comme paramètre. Elle retourne un array avec les centres et un autre avec les rayons.

3.12 sift(image) :

reçoive une image et retourne ses points clés SIFT. Plus précisement, elle retourne trois variables : image qui représent l'image mais en format double, un array de descriptors dont chacun a 128 columnnes et aussi elle retourne une matrice representant la localization de chaque descripteur dans l'image.

Références

1. Cours traitement d'image <https://mootse.telecom-st-etienne.fr>
2. MathWorks <https://it.mathworks.com/>