

Projet d'analyse d'image

Détection et reconnaissance de panneaux de signalisation routière

CARLOS VALADARES , GABRIELLA BETTONTE

28 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 nous a semblé légitime car, puisque nous sommes 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

Ensuite, on a transformé l'image RGB en une image RGB normalisée. Pour cela, nous avons écrit la fonction matlab *RGBNormalize* qui a pour paramètre d'entrée l'image originale et qui renvoie l'image normalisée. Au début, nous avions simplement essayé de diviser chaque canal de couleur par 255, mais cette idée a donné de mauvais résultats. Nous avons donc constaté que la normalisation donnant de bons résultats



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

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.



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 $GMeanThreshold$

2.4 TO BE FIXED :

Ici, je ne me souviens pas bien parce que nous avons fait BWAREAOPEN et comment nous avons extrait le fonds. apres on a fait le canny : il faut expliquer

+IMAGE FOND

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 expansion avec un élément structurant de disque afin de fermer les contours pour délimiter parfaitement la zone souhaitée.

FILLED POUR REMPLIR LE TROUS+ IMAGE

2.6 bounding box

BOUNDING BOX ET EXPLIQUER LE CONDITIONS.+IMAGE

TO BE FIXED :

3 blabla

HSV :

- pourquoi hsv
- comme

probleme : Problem avec thresholding solution : color segmentation app

problem : plaque loin ou peu illumine solution : RGB normalized (pour avoir plus de performance)

4 idee future

- essayer avec filtre gaussian avant segmentation
 - S entre 0.2-0.5
 - ouverture
 - bwareaopen
- NB pas obligé de faire un segmentation parfaite

5 TP 11 janvier

- nous avons essayé de calculer la luminosité avec cette formule $Y = 0.2126R + 0.7152G + 0.0722B$ mais dans les images normalisées, il y avait toujours la même valeur 0.3333
 - au lieu de cela, dans les images, rgb n'était pas une donnée utile car nous ne pouvions pas détecter de motif parmi les images mal détectées
 - après on a fait image original- bwareaopen()
 - img.23-> problème de détection
 - img.16-> il a beaucoup de pixels
 - toutes les autres pas détectées : problème de connexité

6 future pt.2

on va essayer de faire une ouverture pour éliminer les petits choses.

7 Etapes principales

7.1 Seuillage

dffhiufdhf

pourquoi et comment

Problems

Solution

7.2 Detection

7.3 Classification

8 Conclusion

8.1 Retrospective

8.2 Idées futures

Références

1. OpenCV Canny's tutorial https://docs.opencv.org/3.4/da/d5c/tutorial_canny_detector.html