Reconnaissance d'objet pour la détection de panneaux de signalisation routière

Introduction

Pour ce projet, nous avons réalisé un système de la détection de panneaux de signalisation routière qui peut lire une image colorée en RGB et retourner la position et le type du panneau de signalisation routière et l'afficher dans une petite fenêtre si elle en contient.

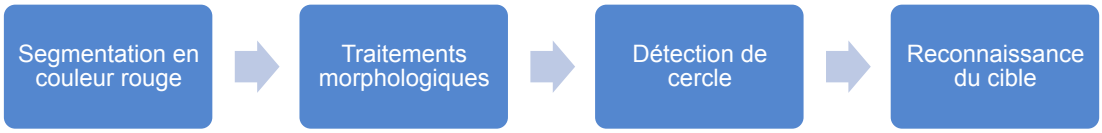
Figure. Image contient un panneau

Pour simplifier le problème, nous plaçons les panneaux à détecter sur les trois suivants : panneau de limitation de vitesse à 30 km/h, panneau de limitation de vitesse à 50 km/h et panneau de signalisation de danger. Nous avons créé une base de données de 41 images pris dans la rue à analyser, parmi lesquelles, il y a ces trois types de panneaux dans différente distances et sous différentes sombres. Il y a aussi des images contiennent des panneaux avec des défauts pour tester l'étendue de cette application système.

Figure. Limitation de vitesse à 30 km/h limitation de vitesse à 50 km/h signalisation de danger

Méthodologie

Pour réaliser la fonction, nous suivons 4 étapes suivantes :



Segmentation

Nous avons tout d'abord besoin de faire une segmentation en rouge afin d'obtenir leurs bords. Dans notre algorithme, nous considérons qu'un pixel est en rouge quand l'index du rouge multiplié par 1.3 est supérieur à la somme de l'index du bleu et du vert. Sur certaines images à analyser, le panneau est dans une condition rétro-éclairage, qui n'ont pas assez de contraste pour détecter le rouge. Pour résoudre ce problème, nous avons augmenté la luminosité dans la zone HSV quand nous détectons une basse luminosité.

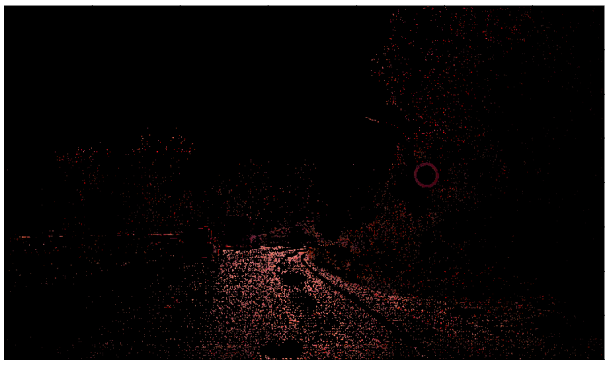


Figure. Image après segmentation

Figure. Panneau avant prétraitement

Figure. Panneau après prétraitement

Traitements morphologiques

Ici, nous avons mis en œuvre une série de traitements morphologiques concernant l'ouverture, la fermeture pour supprimer des défauts, le remplissage de troue et la segmentation par ligne de partage des eaux afin de préparer l'image pour la prochaine étape, la détection de forme.

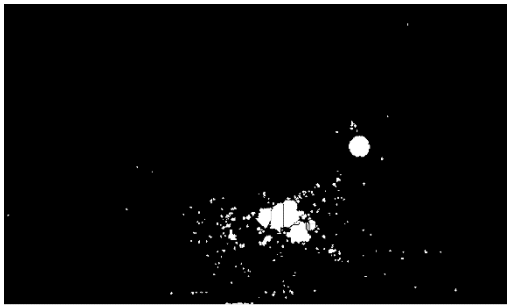


Figure. Image après traitements morphologiques

Détection d'objet circulaire

Nous avons d'abord mesuré les propriétés des régions d'image et obtenu les centres de gravité, les longueurs des axes long et court des zones. En filtrant la longueur des axes long et court et leurs différences de chaque zone, des objets en forme de polygone régulier sont sélectionnés et leurs centres de gravité et leurs rayons estimés sont enregistrés.

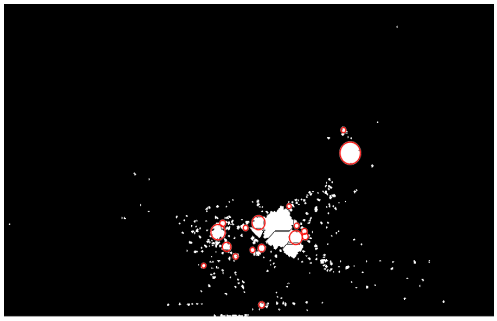


Figure. Image après détection d'objet circulaire

Détection de panneaux

Ce module est divisé en deux parties. Dans la première partie, nous avons prétraité les trois références de panneaux, les avons convertis à la même taille et convertis en images binaires. Selon les rayons et les centres des zones cibles obtenues dans la section précédente, nous pouvons extraire les zones de l'image d'origine et les convertir en des images binaires après prétraitement des couleurs et redimensionnement. Nous comparons ensuite les images binaires à tester avec les trois références d'images binaires, comptons le nombre de pixels ayant la même valeur à la même position, comparons les données avec le nombre total de pixels. Le résultat est donc le taux de coïncidence. Chacun des trois exemples peut obtenir un ensemble taux de coïncidence de blocs. Dans la deuxième partie, nous filtrons le bloc avec le taux de coïncidence le plus élevé dans chaque groupe et ajustons les parties rouge et grise en blanc, c'est-à-dire essayons de ne conserver que la partie noire, puis effectuons la même opération pour les trois références d'image. Effectuez ensuite la comparaison et calculez le taux de coïncidence. La position du bloc avec le taux de coïncidence le plus élevé et son contenu seront affichés dans l'image d'origine.

Figure. Image après détection de panel de signalisation routière

Interface (GUI)

Etape de l'utilisation :

Saisir une image → détecter

Sur notre interface, nous avons réalisé la fonction de saisir et afficher l'image originale, après la procédure de détection, si elle contient un panneau, le programme peut le trouver et l'afficher dans la petite figure à droite en précisant sa position et son type dans les text-boxs en bas.

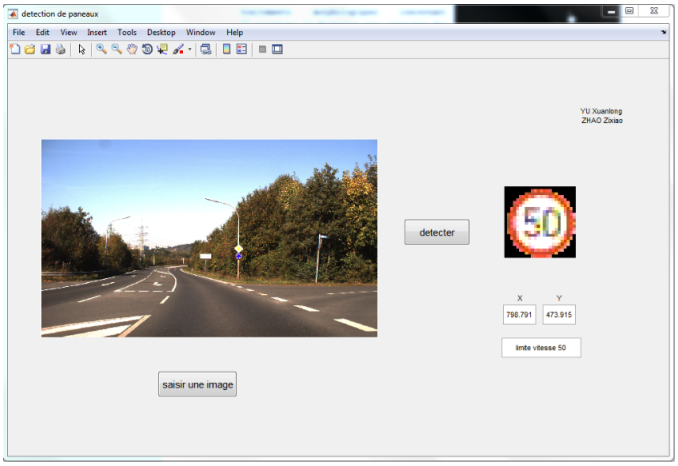


Figure. GUI avce un test de détection

Analyse du résultat

A base de notre jeu de données sur 41 images (les images sur mootse), nous pouvons bien détecter les panneaux sur 32 images, le taux de réussite est de 78.04%. Mais pour la reconnaissance du type de panneau, notre taux de réussite est à 48.78% (20 images).

Notre algorithme est bien adapté pour les conditions en commun, mais il y a encore des problèmes quand le panneau est sous une condition de bas contraste.

Et aussi même nous avons essayé plusieurs façons pour améliorer la reconnaissance du type de panneau, nous n'avons pas suffisamment réussi dans cette partie.

Explication des fonctions

detection.m

I_seg = detection(I_in);

Cette fonction sert à segmenter l'image d'entrée I_in à base de la profondeur du rouge, c'est-à-dire de retirer les objets rouges dans l'image originale. Donc dans l'image de sortie, ces objets sont conservés en rouge et le fond est transféré en noir.

I_in : l'image d'entrée en couleur RGB sous format uint8.

I_seg : l'image de sortie codée en rouge et noir sous format uint8.

morph.m

I_bw = morph(I_seg);

Cette fonction sert à réaliser une série de traitements morphologiques sur l'image segmentée afin de supprimer des défauts (petite morceaux, trou sur les objets et etc.) et séparer les objets connectés. Cette étape peut considérée comme un pré-traitement pour la détection de forme.

I_seg : l'image d'entrée segmentée sous format uint8.

I_bw : l'image de sortie codée en binaire qui ne conserve que les informations de taille, forme et position des objets.

detectionCircle.m

[circleElement,I_tire] = detectionCircle(I_bw,I);

Cette fonction est conçue pour détecter des cibles circulaires (triangles) dans une image binaire en notant leurs diamètres et leurs positions et aussi en retournant une image I_tire qui ne garde que le couleur de la zone du cercle qui vient d'être détectée sur l'image originale, et changer le fond en noir.

I_bw : l'image d'entrée codée en binaire.

I : l'image d'entrée en couleur RGB sous format uint8.

circleElement : une matrice n*3 qui conserve les diamètres et les coordonnées x et y des cercles qu'on a détecter sur l'image I_bw.

I_tire : l'image de sortie qui ne garde que le couleur de la zone du cercle qui vient d'être détectée sur l'image originale.

detectionPanel.m

[content, centerx, centery] = detectionPanel(I,circleElement);

Cette fonction sert à retirer les objets circulaires (triangles) correspondants aux informations obtenues par la fonction detectionCircle et puis les comparer avec les motifs (base_ref) afin de justifier leurs types.

I : l'image d'entrée en couleur RGB sous format uint8.

circleElement : la matrice n*3 qui conserve les diamètres et les coordonnées x et y des cercles que nous avons obtenu.

centerx, centery : les coordonnées x et y du panneau que nous trouvons au finale.

content : le type du panneau détecté.

Référence

- Aide Matlab
- https://fr.mathworks.com/matlabcentral/answers/47058-how-to-set-value-in-edit-text-box L'utilisation de text-box sur GUI
- http://www.ilovematlab.cn/thread-29736-1-1.html Détection de forme