TWIZZY

Détecteur de panneaux de signalisation

MISE EN SITUATION

Les constructeurs automobiles proposent aujourd'hui des technologies dites anti-excès de vitesse permettant de lire automatiquement les panneaux de limitation de vitesse sur le bord des routes afin d'intégrer cette information sur le tableau de bord et/ou de la coupler avec les limiteurs automatiques de vitesse.

Ces systèmes ont commencé à être présents sur des véhicules haut de gamme (Ford S-MAX, Mercedes Classe S, Audi A8) et se démocratisent aujourd'hui sur un parc élargi de véhicules grand public.

*Figure 1:Le système « **Traffic Sign Assist »** de Mercedes est intégré au véhicule*

Figure 2:Caméra multifonctions permettant la détection de panneaux de signalisation (BOSCH MPC2)

Dans ces systèmes une caméra logée dans le haut du pare-brise, ou dans le rétroviseur permet de filmer l'avant du véhicule. La caméra fait partie intégrante du réseau de capteurs d'environnement et est utilisable avec les autres capteurs, tels que les capteurs radar et les capteurs à ultrasons.

Par ailleurs, dans un tout autre registre, il existe aussi des applications Android et iOS, qui fonctionnent également de manière autonome sur un smartphone attaché au pare prise du véhicule. Ces applications permettent aussi d'informer le conducteur sur la limite de vitesse en cours.

*Figure 2: L'application « **myDriveAssist** » de l'équipementier Bosch*

Tous ces systèmes sont basés sur un noyau logiciel de traitement d'images qui est capable de détecter et d'isoler les formes rondes caractéristiques des panneaux de signalisation pris en charge et même de détecter les panonceaux associés.

Le système qui fait l'objet de ce projet Twizzy est composé d'une caméra, d'un noyau logiciel réalisant cette détection et d'une base de données. Le noyau est codé en Java et utilise la bibliothèque Open CV 2.4.9. La base de données est MySQL 5.7

Ce noyau sera d'abord testé en analysant des images fixes, puis en analysant en temps réel un flux vidéo.

Travail demandé :
La réalisation de ce travail expérimental nécessitera la mise en œuvre des connaissances et des compétences suivante :

3.2. Programmation par objets

- Concepts de base d'un langage objet (classe, attribut, méthode, etc.)
- Utilisation de bibliothèques de composants
- Interfaces graphiques
- Programmation événementielle.

3.6. Principes des bases de données

- -Définition d'un schéma relationnel en S.Q.L ;
- -Accès à une base de données depuis un langage de programmation ;

3.7. Traitement d'images

- Introduction aux images numériques.
- Filtrage 2D
- Détection de contours.
- Introduction aux images couleurs.

PREMIERE PHASE : PREMIERE PARTIE

Présentation des fonctionnalités proposées par le noyau :

Les différentes étapes de la détection de panneaux sur une image fixe sont présentées ci-dessous.

Passage du codage de l'image de l'espace Bleu Vert Rouge (BGR) à l'espace Teinte Saturation Valeur (HSV)

Saturation des rouges

Détection de contours

Une analyse de forme est ensuite effectuée sur les contours obtenus. Seuls les contours en forme de cercles, triangles et rectangles sont retenus.

Les images incluses dans ces contours sont ensuite extraites de l'image originale. Sur ces extraits d'images sont calculés des points d'intérêts ainsi que leurs descripteurs associés grâce à la méthode ORB (Oriented, Fast and Rotated BRIEF), cette méthode est basée sur une variante de FAST (pour les points d'intérêt) et de BRIEF (pour les descripteurs). Les descripteurs d'ORB sont des vecteurs de 32 octets.

Figure 3 : A gauche le panneau extrait de l'image, à droite le panneau de référence. Les descripteurs calculés sont ensuite matchés les uns aux autres.

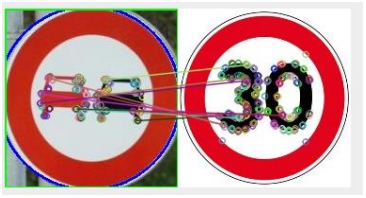


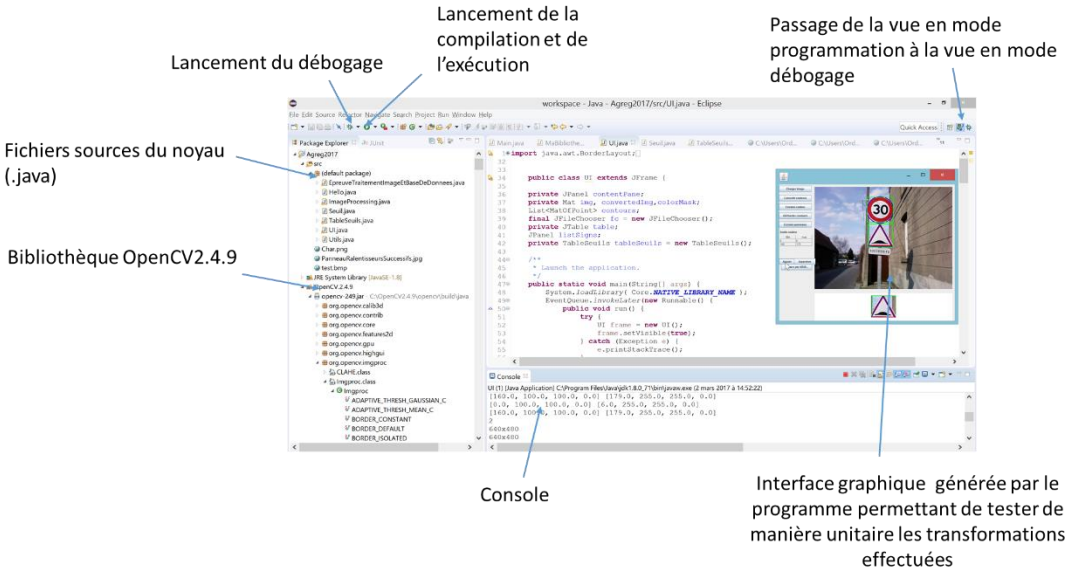
Figure 4: Même si le panneau extrait est diffèrent de la référence, des similitudes peuvent être détectées parmi les détecteurs

Ces descripteurs sont comparés à une base de descripteurs pré-calculés sur des panneaux de référence. Ces descripteurs pré-calculés sont stockés en base de données.

Un critère de similitude est enfin calculé entre le panneau filmé et tous les panneaux de référence.

Présentation de l'interface de développement:

L'interface de développement du noyau de traitement d'image *Eclipse Néon 1.a*



PREMIERE PHASE :

Première étude :

Objectif : Extraction des pixels rouges et création d'une image saturée

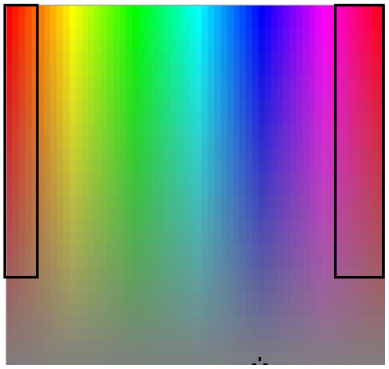
1. Décomposer l'image fournie en un vecteur de matrices de type <*Vector*>*Mat*. Le vecteur contiendra les matrices correspondant aux 3 composantes H,S et V. La fonction ***core.split(Mat m,List<Mat> mv)*** pourra avantageusement être utilisée.

2. Créer des objets de type ***Scalar*** dans lesquels seront stockées les valeurs des seuils qui définiront les limites de la couleur rouge.

3. Pour chaque pixel, effectuer les opérations de comparaison par rapport aux seuils précedemment fixés. Saturer les pixels choisis à la valeur 255. Les fonctions ci-dessous pourront avantageusement être utilisées :

   void Core.compare(Mat src1, Scalar src2, Mat dst, int cmpop)
   void Core.bitwise_or(Mat src1, Mat src2, Mat dst)
   void Core.bitwise_and(Mat src1, Mat src2, Mat dst)

4. Appliquer ce traitement à l'image témoin ci-dessous (fichier ***temoin.jpg*** présent sur le bureau.). Régler approximativement les seuils pour ne conserver que les rouges contenus dans les rectangles noirs.



5. Appliquer un filtre gaussien pour flouter l'image et lisser les imperfections. Afficher l'image obtenue en appliquant la chaine de traitement sur l'image ***panneau0.jpg***

Deuxième étude :

Objectif : Se constituer une base de données de descripteurs pour les panneaux de réference

Dans cette partie, il va être demandé aux étudiants de pré-calculer les descripteurs des panneaux de référence et de les stocker dans une base de données. Dans le cadre de cette étude, seuls les 12 panneaux de limitation de vitesse classiques seront analysés et stockés.

6. Lancer la méthode ***extractDescriptor()***. Celle-ci a été didactisée pour afficher dans la console les differentes étapes de son éxécution. Afficher dans la console pour un panneau choisi, l'ensemble des points d'interêts (keypoint) calculés ainsi que leurs coordonnées.
7. On s'interessera maintenant principalement à la methode ci-dessous :

   void features2d.DescriptorExtractor.compute(Mat image, MatOfKeyPoint keypoints, Mat descriptors)

   A l'issue de son exécution la matrice ***Mat descripitors*** contient les déscripteurs associés à chaque point clé. Afficher dans la console pour un point clé donné les descripteurs associés (un decripteur=32 octets).
8. A l'aide de mySQL Workbench dessiner une base de données permettant de stocker les informations suivantes :
   a. Chaque panneau analysé doit avoir un nom (panneau20, panneau30 etc);
   b. Pour chaque panneau une collection de points d'interêts doit être enregistrée. Il n'y pas toujours le même nombre de points d'interêt par panneau. Les points d'interêt sont décrits à minima par une abscisse et une ordonnée ;
   c. A chaque point d'interêt est associé 32 descripteurs (1 decripteur = un octet).
9. Créer cette base de données et l'enrichir avec des données relatives aux 12 panneaux (nom, points d'interêts, descripteurs associés).

Objectif : Intégrer du code et réaliser la détéction sur un flux vidéo.

10. La methode ***ReconnaissancePanneau()*** utilise la fonction ***match()*** d'openCV et permet de mettre en liens des points d'interêts via les informations contenues dans les descripteurs. Cette méthode didactisée permet de comparer un panneau extrait à tous les panneaux de la base de données. Elle renvoie un score de ressemblance pour chacun des panneaux.

La méthode didactisée ***AnalyseVideo()*** permet de lire un flux vidéo , d'en extraire les images, de les convertir en Matrice OpenCV et donne un exemple élementaire de traitement d'images (comparaison d'images, détection de points).

En s'inspirant de ces 2 méthodes didactisées, écrire un programme qui lit la vidéo fournie, et qui affiche en temps réel dans la console la valeur de la dernière limitation de vitesse rencontrée.