Detection des places occupées

La détection des places occupées à partir du flux vidéo d’une camera est réalisé grace à l’API OpenCV. Pour des raisons de performances le programme de détection est développé en c++ sous la forme d’une bibliothèque appelée depuis un programme en JAVA ( appel JNI ).

Il s’agit de détecter dans le flux vidéo les éléments en mouvement et de déterminer leur baricentre. Ensuite on retourne le résultat à l’application java appelante qui se charge des autres traitements.

# 1°)Detection des obstacles en mouvement

**(methode positionDesBlobs)**

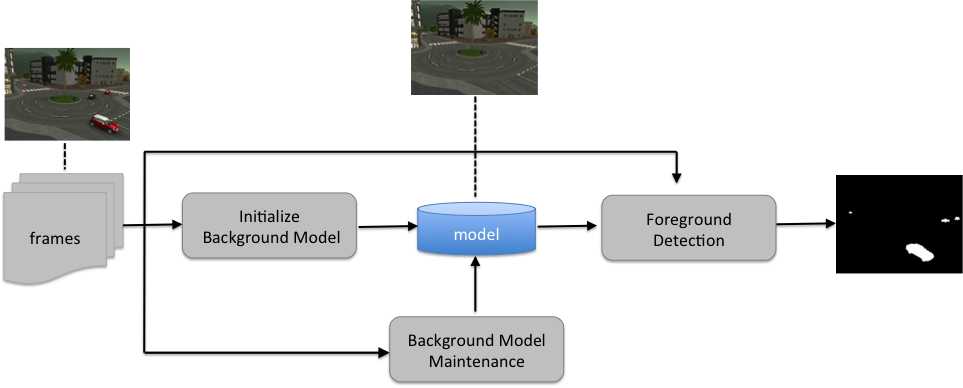
Un parking étant un environnement relativement stable une méthode sûre et performante consiste à comparer d’une image à l’autre ce qui est fixe et ce qui à bougé entre l’image courante et une image de référence, c’est le principe des algorithmes de la famille des BackgroundSubstractor.

Si on fait une simple soustraction d’images entre l’image précédente et suivante on obtient bien la différence entre les deux images mais avec énormément de bruit ( faux positif ).

pour accelerer le traitement on travaillera sur une image réduite.

Ici on utilisera l’algorithme MOG2 qui donne déjà de bon résultats.

Le principe est celui-ci :



On initialise l’arrière plan, on compare l’image suivante par rapport au modèle d’arrière plan et en sortie on obtient un masque avec la différence entre les deux images puis on met à jour le modèle d’arrière plan. Des paramètres permettent de définir la durée d’immobilité d’un sujet avant d’être intégré à l’arrière plan et le flou à appliquer pour réduire le bruit.

En sortie de l’algorithme on obtient la silhouette des objets en mouvement et toujours quelques bruits.

En suivant on cherche à regrouper les zones par objets ( blobs ) en délimitant des contours ( connexion entre les points ) ce qui nous permet d’une part d’écarter les objets de petite taille ( en fonction de la surface inférieure à un seuil ) et de déterminer le baricentre de l’objet.

A partir d’ici les traitements s’effectuent coté JAVA

# 2°)Calcul de l’état de la place

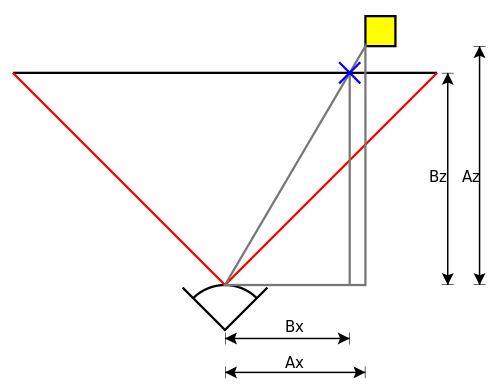
A chaque retour de la méthode C++ positionDesBlobs on calcule pour chaque blob sa position par rapport aux places visibles pour déterminer si une place est occupée ou se libère.

# 2.1°)Calcul des places visibles sur une caméra

Pour une caméra donnée il faut d’une part déterminer les places visibles sur l’image par rapport au plan et d’autre part déterminer ou se trouve le repère de la place de parking sur l’image par rapport à la réalité.

Il s’agit, pour trouver les places visibles, de calculer le champ de vision de la caméra en fonction de sa position sur la carte, son orientation et sa focale.

Pour celà on va appliquer des calculs de projection.



On utilise la formule pour passer des distance réelles aux distances sur l’image.

La distance Bz est donnée par la focale ( pour une caméra c’est la distance entre la lentille et le récepteur ).

En appliquant cette formule aux informations fournies par le serveur il est possible de calculer si un objet se trouve dans le champ de vision de la caméra.

A partir de là on compare les positions de blobs avec les positions des repères des places visibles et on obtient les places changeant d’état.

# 2.2°)Calcul du nouvel état

A partir des positions du blob ( on a les angles formant le cadre contenant le blob ) et de la position du repère des places de parking on est capables de dire si le repère est inclus dans le périmètre du blob. Si le blob est sur le repère et que la place était vide c’est que la place est maintenant occupée.

Si la place est occupée et qu’un blob apparait dessus on le suis jusqu’à ce qu’il disparaisse. S’il disparait alors qu’au test précédent il y avait un blob sur une place occupée c’est que le sujet s’en va, la place est donc libre.

3°) Communication avec le serveur

Les places étant souvent visibles par plusieurs caméras on compare leur état avant de le mettre à jour.

La communication se fait par des API REST par lesquelles on échange des objets JSON.

Elle se limite à récupérer la configuration au lancement de l’application et renvoyer l’état des places lors de leur changement d’état.