# TP2 : Détection de mouvement dans les vidéos

## EXERCICE 1 : ALGORITHME DE SEGMENTATION PAR SEUILLAGE SIMPLE

### Question 1.1

Pour ouvrir une vidéo, on utilise la classe « VideoCapture » d’OpenCV.

    VideoCapture video("video/111.mp4");

    if (!video.isOpened()) {

        std::cerr << "Error opening video stream or file" << std::endl;

        exit(EXIT\_FAILURE);

    }

Pour accéder aux images de la vidéo, on fait une boucle qui nous permet de récupérer chaque frame une à une dans un objet de type « Mat ».  
On peut ensuite les afficher simplement.

while (true) {

        Mat frame;

        // On recupere une frame

        video >> frame;

        // On vérifie qu'on est pas a la fin de la video

        if (frame.empty())

            break;

        // On affiche la frame

        imshow("Frame", frame);

        // On ferme la video lors de l'appuie sur la touche ESC

        char c = (char)waitKey(25);

        if (c == 27)

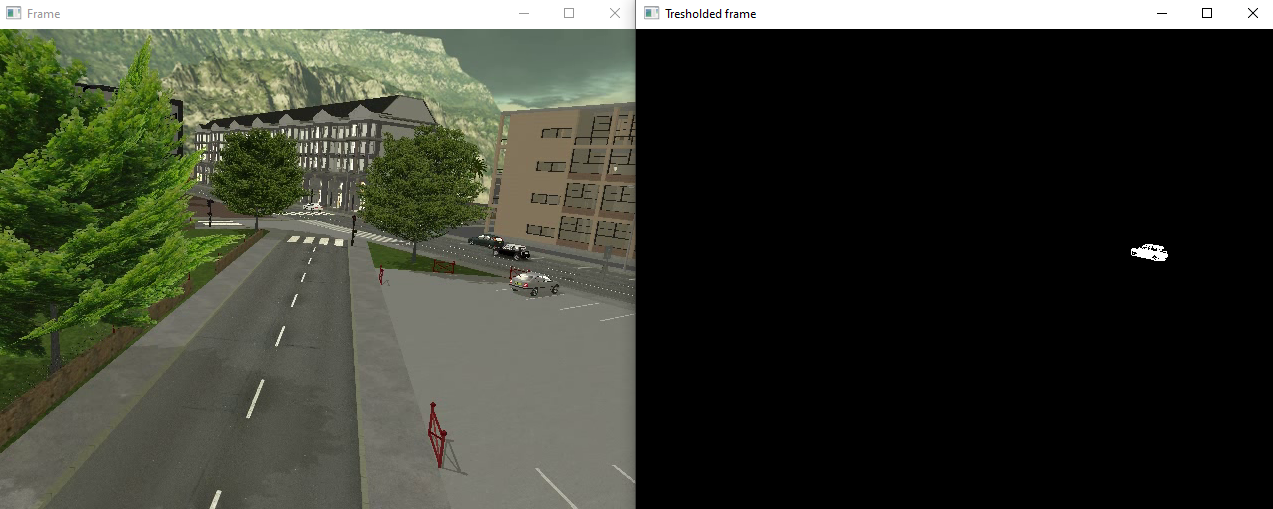
            break;

    }

    // On ferme la video

    video.release();

### Question 1.2

Pour mettre en place l’algorithme de seuillage, j’ai réalisé une fonction qui prend en paramètre l’image de référence, l’image à seuiller ainsi que la valeur du seuil T, et qui retourne l’image seuillée. Ensuite, on appelle cette fonction dans notre boucle, et on affiche le résultat. Un seuil T égal à 40 semble donner un résultat correct.

## EXERCICE 2 : MISE EN PLACE D’UN MODÈLE GAUSSIEN POUR LA DÉTECTION DE MOUVEMENTS

### Question 2.1

Pour changer le nombre de distribution gaussiennes dans le modèle de fond, il faut utiliser la fonction « setNMixtures() ».

On connait aussi la fonction « setVarTreshold », qui permet de choisir la valeur du seuil T.

### Question 2.2

Pour mettre en place le modèle à plusieurs distributions gaussiennes, on utilise la fonction « createBackgroundSubtractorMOG2() » d’OpenCV. Ensuite, pour utiliser plusieurs distributions gaussiennes, on utilise la méthode « setNMixtures() » de l’objet BackgroundSubtractorMOG2 qui prend en paramètre le nombre de distributions que l’on veut utiliser.

### Question 2.3

Pour calculer les temps d’exécutions des deux méthodes, j’ai choisi d’utiliser la bibliothèque chrono. Ensuite, je récupère le temps actuel avant et après l’appelle des fonctions. Puis, lorsque que la vidéo s’arrête (à la fin ou lors de l’appuie sur la touche échap), on calcule la moyenne du temps d’exécutions. On se rend compte que la méthode d’OpenCV est plus efficace que la méthode par seuillage simple que nous avons implémentée. On se rend aussi compte que le temps d’exécution de la méthode d’OpenCV est de plus en plus longue lorsque l’on augmente le nombre de distributions gaussiennes.

