```
///<<<<-------------------------------------------------------------------------->>>///
/************||     == DETECT traffic Que based on Vehicle movements ==      ||**************

This code using the princeple of Image processing to calculate the number of vehicles in real time,
using IP camera or
fixed video file.
*/



#include <opencv2/core/core.hpp>
#include <opencv2/highgui/highgui.hpp>
#include <opencv2/video/video.hpp>
#include <stdio.h>
#include <iostream>
#include <vector>
using namespace cv;
using namespace std;
std::vector<std::vector<cv::Point> > contours;
vector<vector<Point> > contours_poly(contours.size());
vector<Rect> boundRect(contours.size());
vector<Point2f>center(contours.size());
vector<float>Distance(contours.size());
vector<float>radius(contours.size());
std::vector<std::vector<cv::Point> > intruders;
Mat fgMaskMOG;
Mat fgMaskMOG2;
Mat frame;
Ptr<BackgroundSubtractor> pMOG;
double keyboard;
void processVideo(char* "C:\Users\seven\Desktop\VD.mp4";
int main(int argc, char* argv[])
{
	pMOG = createBackgroundSubtractorKNN();
	processVideo(0);
}
void processVideo(char* "C:\Users\seven\Desktop\VD.mp4")
{
	VideoCapture capture("C:\Users\seven\Desktop\VD.mp4");
	while (1)
	{
		bool bSuccess = capture.read(frame);
		pMOG->apply(frame, fgMaskMOG);
		cv::Mat kernel, e;
		cv::getStructuringElement(cv::MORPH_RECT, cv::Size(10, 10));
		cv::morphologyEx(fgMaskMOG, e, cv::MORPH_CLOSE, kernel, cv::Point(-1, -1));
		cv::findContours(e.clone(), contours, CV_RETR_EXTERNAL, CV_CHAIN_APPROX_SIMPLE);
		for (int i = 0; i < contours.size(); i++)
		{
			double area = cv::contourArea(contours[i]);
			if (area > 70 && area < 700)
				intruders.push_back(contours[i]);
		}
		cv::Mat mask = cv::Mat::zeros(frame.size(), CV_8UC3);
		cv::drawContours(mask, intruders, -1, CV_RGB(255, 255, 255), -1);
		imshow("B", mask);
		cv::Mat mask2 = cv::Mat::zeros(mask.rows + 2, mask.cols + 2, CV_8U);
```

```
            cv::floodFill(mask, mask2, cv::Point(0, 0), 255, 0, cv::Scalar(), cv::Scalar(), 4 +
(255 << 8) + cv::FLOODFILL_MASK_ONLY);
            erode(mask2, mask2, Mat());
            rectangle(mask2, Rect(0, 0, mask2.cols, mask2.rows), Scalar(255, 255, 255), 2, 8, 0);
            Mat copy;
            mask2.copyTo(copy);
            vector<Vec4i> hierarchy;
            findContours(copy, contours, hierarchy, CV_RETR_TREE, CV_CHAIN_APPROX_SIMPLE, Point(0,
0));
            Mat drawing = cv::Mat::zeros(mask2.rows, mask2.cols, CV_8U);
            int num_v = 0;
            for (int i = 0; i < contours.size(); i++)
            {
                  approxPolyDP(Mat(contours[i]), contours_poly[i], 3, true);
                  if (contourArea(contours_poly[i]) > (mask2.rows * mask2.cols / 10000) &&
contourArea(contours_poly[i]) < mask2.rows* mask2.cols * 0.9)
                  {
                        boundRect[i] = boundingRect(Mat(contours_poly[i]));
                        minEnclosingCircle((Mat)contours_poly[i], center[i], radius[i]);
                        circle(drawing, center[i], (int)radius[i], Scalar(255, 255, 255), 2, 8,
0);
                        rectangle(drawing, boundRect[i], Scalar(255, 255, 255), 2, 8, 0);
                        num_v++;
                  }
            }
            cout << "NUMBER OF VEHICLES  = " << num_v << endl;
            keyboard = waitKey(3);
      }
      capture.release();
}
```