электронный журнал
МОЛОДЕЖНЫЙ НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК
Издатель ФГБОУ ВПО "МГТУ им. Н.Э. Баумана". Эл No. ФС77-51038.

УДК 004.93'12

Алгоритм распознавания дорожных знаков

***Романов П. В.**, студент*
Россия, 105005, г. Москва, МГТУ им. Н. Э. Баумана,
кафедры «Компьютерные системы и сети»

Научный руководитель: Самарев Р. С., к.т.н., доцент
Россия, 105005, г. Москва, МГТУ им. Н. Э. Баумана
petr.2@bmstu.ru

Введение

Цель данной работы – решение задачи распознавания дорожных знаков, а также проверка эффективности и удобства использования алгоритмов, реализованных в библиотеке openCV. Для получения исходных данных применялась камера, закрепленная в автомобиле — типовой видеорегистратор, обеспечивающий видеосъемку в формате Full-HD. Основными задачами были: фиксирование знака на видео с камеры, определение его типа по упрощенной классификации.

Алгоритм распознавания знаков

В процессе анализа возможных подходов рассматривались как «нейронные сети», так и статистические подходы. Библиотека OpenCV реализует большое количество функций, позволяющих реализовать эти подходы по заранее приготовленным шаблонам. Однако в данном случае было принято решение об использовании точных методов, которые также реализованы в OpenCV. Для решения задачи была написана программа-распознаватель. Процесс распознавания осуществлялся в три этапа:

- фиксация в кадре видеопотока области поиска и самого объекта (знака, в случае правильного срабатывания);
- анализ параметров найденного объекта, отнесение его к одному из типов (шаблонов из базы данных);
- адаптация программы для работы в более неблагоприятных условиях: в темноте, присутствии помех.

На каждом из указанных этапов вызываются функции из библиотеки. Обобщенная схема алгоритма анализа приведена на рисунке 1. В схеме показаны этапы и используемые функции OpenCV.

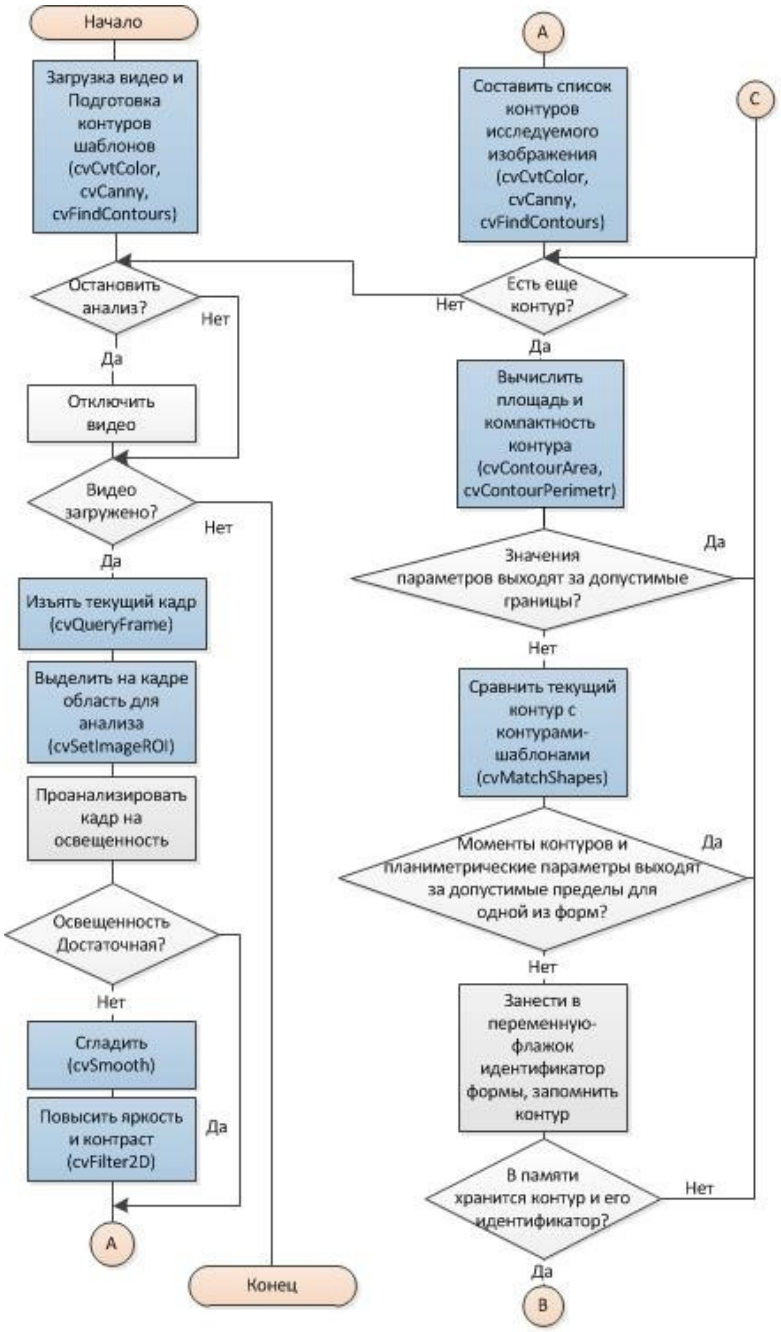

Рис. 1. Обобщенная схема алгоритма (начало)

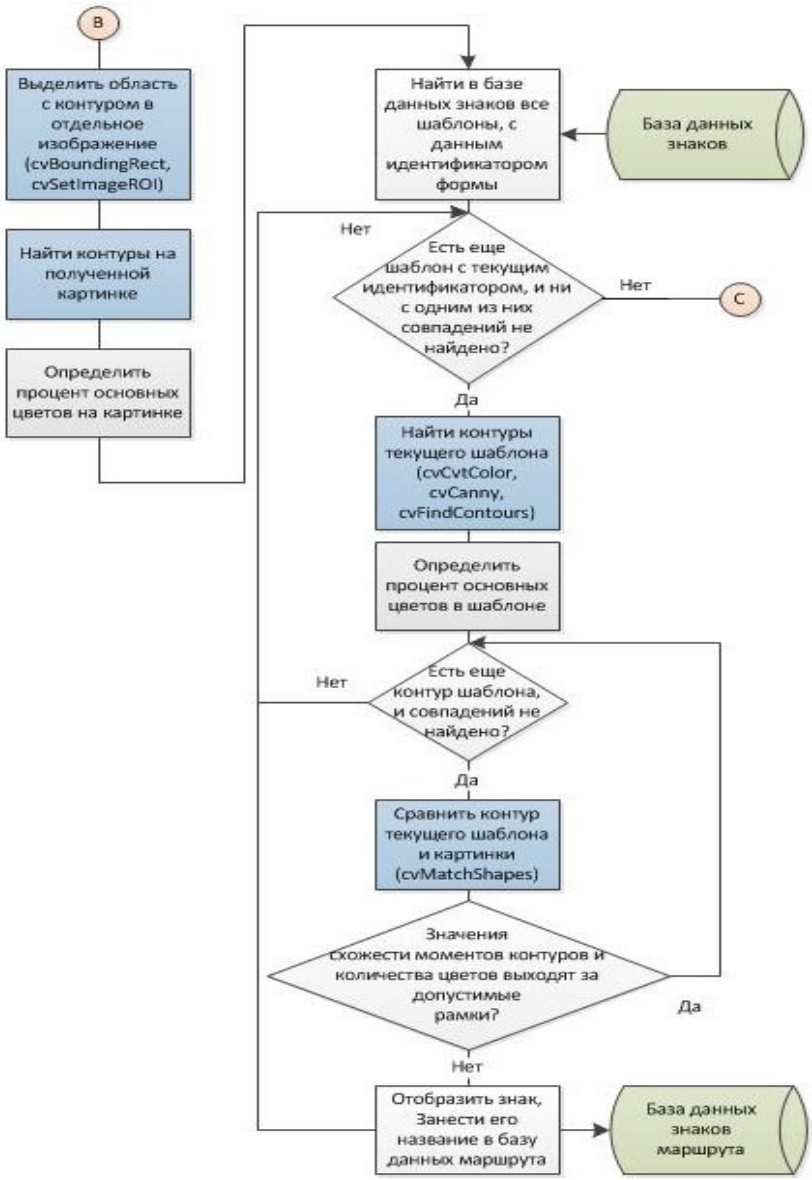

Рис. 1. Обобщенная схема алгоритма (продолжение)

На первом этапе использованы функции cvSetImageROI, cvBoundingRect и cvCopy. Первая позволяет “сфокусироваться” на интересующем участке изображения (ROI – region of interest)[2]. На рисунке 3 можно увидеть, что программа реально анализирует и представляет пользователю лишь часть кадра (рисунок 2). Одни из ее параметров, координаты выделяемого участка, легко вычисляется, поскольку в OpenCV предусмотрено сопровождение картинки необходимой информацией.

Рис. 2. Исходное изображение

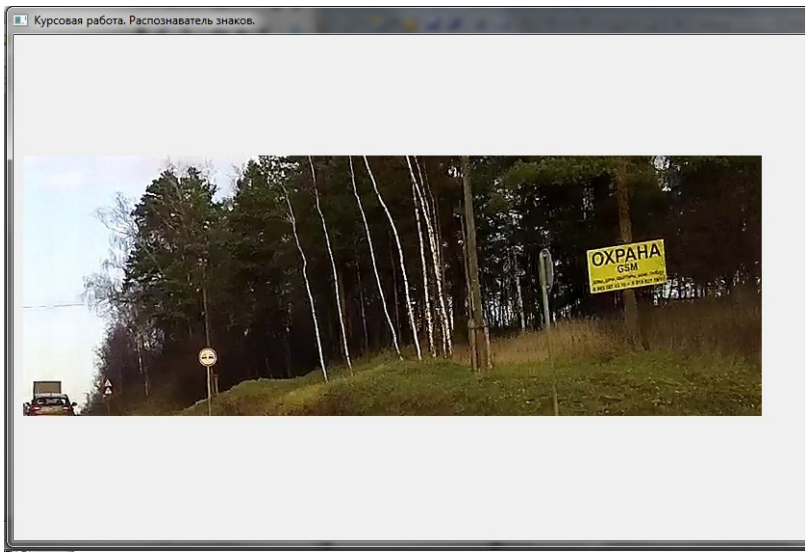

Рис. 3. Анализируемая часть изображения (кадра)

После выделения области, в которой наиболее вероятно появление знаков, примерно средняя треть кадра, нужно приготовить ее для поиска похожих на знаки форм. Для этого использовались следующие инструменты из openCV. Функция cvCvtColor с опцией перевода картинки в градации серого, необходимая для применения порогового преобразования (функция cvThreshold) или преобразования Кенни (функция cvCanny)[3]. Оба метода предназначены для бинаризации изображения(только черное и белое) и выделения на нем границ объектов, но первый учитывает только яркость участков изображения, тогда как алгоритм Кенни также выполняет частичное восстановление контуров. Эксперименты подтвердили преимущество функции cvCanny (рисунок 4).

Рис. 4. Результат последовательного применения cvCvtColor и cvCanny к изображению

Далее на изображении нужно найти все возможные замкнутые контуры. Для этого использована функция cvFindContours, имеющая параметром бинаризированное изображение, полученное ранее[4]. Были попытки применить оператор Собеля для вычисления градиента яркости каждой точки картинки (функция cvSobel) и путем преобразования Хафа найти геометрические фигуры, в данном случае круги (функция cvHoughCircles). Ни одна из этих функций не показала себя в экспериментах достаточно универсальной в силу неточности комбинаций входных параметров при работе с видео.

Следующий шаг – выбор контуров, удовлетворяющих заданным условиям. Первое, что анализировалось – площадь контура, поскольку она колеблется в заранее известных границах[2]. Ее можно получить, вызвав cvContourArea. cvContourPerimeter возвращает длину контура, что необходимо для вычисления второго параметра – компактности. Это отношение площади ко квадрату периметра. Проще говоря, характеризует схожесть объекта с кругом, поскольку круг – самая компактная фигура. – имеет коэффициент примерно 0,79. И наконец, проводилась проверка на совпадение моментов контуров (их отличительных форм) с помощью функции openCV cvMatchShapes, сравнивающей переведенные в цепной код Фримана контуры и выдающей уровень их отличия. Опыты показали, что почти полное сходство выражается значением меньше 0.06 – 0.08. На рисунке 5 – пример всех зафиксированных контуров (выделено ярко зеленым и фиолетовым), а на рисунке 6 – правильно узнанных знаков.

Рис. 5. Все найденные в области контуры

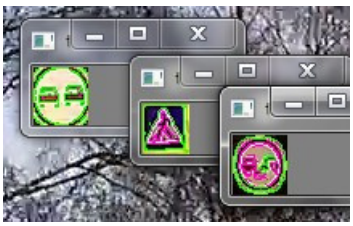

Рис. 6. Контуры, подошедшие по форме

По результату работы этой функции бывает нельзя точно сказать, подходит контур или нет, поэтому последний из приемов установления различий – это анализ цвета в контуре. Для разбиения изображения на цветовые каналы применялись функции cvSplit и cvCvtPixToPlane. Однако функция cvCvtPixToPlane предназначена для изображений, представленных не цветовыми, а тональными каналами: оттенком, яркостью и глубиной (модель HSV)

Отличие второго этапа от первого в меньшем размере обрабатываемого изображения, но более тщательном сравнении моментов контуров. Происходит сопоставление со всеми знаками базы данных, удовлетворяющими условиям, проверенным на предыдущем этапе. Также производилось вручную контрастирование изображения для обеспечения более четкой работы функций детекции границ.

Результаты

Выборка сформирована с учетом включения различных форм знаков и изображений на них. Тестировалось распознавание 7 знаков на 16 видео фрагментов с разрешением 1920x1080. Использовано видео в ясный день. Результаты сведены в таблицу.

Пример знака	Точность распознавания		Пример формы знака	Точность распознавания
Обгон запрещен	85%		Круглый	75%
Въезд запрещен	65%			
Ограничение скорости 40 км/ч	65%			
Ограничение скорости 70 км/ч	70%			
Поворот запрещен	40%			
Уступите дорогу	70%		Треуголь-ный	75%
Примыкание второстепенной дороги	80%			

Заключение

По результатам проведенной работы можно отметить возможность использования библиотеки OpenCV для решения задачи распознавания дорожных знаков. Распознается примерно 70%, что, однако, говорит о необходимости нахождения более оптимальных методов.

В качестве возможного направления работ также следует отметить исследование распознавания знаков в сложных условиях, например, ночью или при тумане.

Список литературы

1. Ворошин Г. Я Методы распознавания образов. Режим доступа: http://abc.vvsu.ru/Books/Metody_r/default.asp (дата обращения 15.04.2014).
2. Владимир Н. OpenCV шаг за шагом. Режим доступа: http://robocraft.ru/page/opencv/ (дата обращения 15.04.2014).
3. Распознавание образов с OpenCV: контуры против haartraining Режим доступа: http://www.pvsm.ru/algoritmy/30704 (дата обращения: 15.04.2014).
4. Gary Bradski, Adrian Kaehler. Learning OpenCV. Computer vision with the OpenCV library. Available at: http://locv.ru/wiki/%D0%93%D0%BB%D0%B0%D0%B2%D0%BD%D0%B0%D1%8F_%D1%81%D1%82%D1%80%D0%B0%D0%BD%D0%B8%D1%86%D0%B0, accessed 15.04.2014.
5. RECOG.RU: распознавание образов для программистов Режим доступа: http://recog.ru/page/library/opencv (дата обращения: 15.04.2014).