УДК 004.421

МЕТОД ПОПИКСЕЛЬНОГО СРАВНЕНИЯ И РАСПОЗНАВАНИЯ ОБЪЕКТОВ

Лаврентьев Е. Б., Кирпа В. Э.
Донской государственный технический университет, Ростов-на-Дону, Российская Федерация
leb210570@gmail.com
kirpavitali@yandex.ru

В статье описывается метод попиксельного сравнения и распознавания объектов, полученных с помощью видеокамеры для распознавания дорожных знаков.

Ключевые слова: распознавание, дорожные знаки, робот, алгоритм, фильтрация, цвет.

UDC 004.421

METHOD OF PIXEL BY PIXEL COMPARISON AND RECOGNITION OF OBJECTS

Lavrentyev E. B., Kirpa V. E.
Don State Technical University, Rostov-on-Don, Russian Federation
leb210570@gmail.com
kirpavitali@yandex.ru

The article describes the method of pixel-by-pixel comparison and recognition of objects obtained using a video camera for road signs recognition.

Keywords: recognition, road signs, robot, algorithm, filtering, color.

Введение. В рамках создания автономного мобильного объекта — робота для движения в условиях, приближенных к городскому трафику, возникает задача однозначного распознавания дорожных объектов — знаков, которые предписывают направление дальнейшего движения робота на перекрестке [1].

Постановка задачи. Таким образом, в результате распознавания необходимо формировать некий флаг, регламентирующий дальнейшее управление системой движения робота. В задаче был применен ограниченный набор основных знаков, которые предполагалось располагать на пути следования робота, а именно: «только вперед», «только налево», «только направо», «только прямо и налево», «только прямо и направо» (рис. 1).

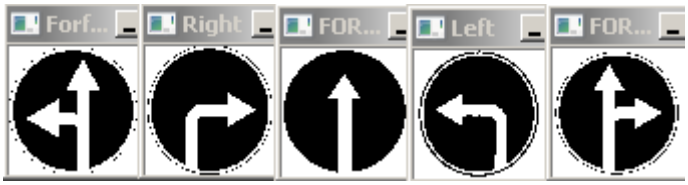


Рис. 1. Знаки для распознавания

Реализация метода. Программное обеспечение для распознавания было создано в среде Visual Code [2]. Изначально для распознавания образов планировался алгоритм, который основывался на том, что сначала распознавали окружность, которая попала в объектив видео камеры, затем эта окружность делилась на 9 фрагментов. Из этих девяти фрагментов выбиралось четыре ключевых, в которых могли находиться белые части знака. Далее использовалась функция get_dominant_color, которая находила доминирующий цвет в этом фрагменте. Например, если белый цвет наблюдался только в верхнем и нижнем фрагментах, то можно было утверждать, что это знак «Прямо». Однако данный подход оказался неприемлемым в реальных условиях движения, поскольку при малейшем отклонении окружность, на которой был изображен знак, «превращалась» в эллипс, ее фрагменты также отклонялись, следовательно, невозможно было распознать белый цвет там, где он был. Особенно сильно это проблема проявлялась у знаков «Только направо» и «Только налево». По этим причинам данный способ был отклонен.

Для написания алгоритмов распознавания знаков была использована открытая библиотека OpenCV. Для достижения поставленной цели были реализованы два алгоритма распознавания:

- метод попиксельного сравнения с эталонным изображением;
- метод распознавания формы.

Метод попиксельного сравнения с эталонным изображением

Данный метод заключает в себе несколько этапов:

- загрузка эталонных изображений знаков;
- преобразование цветового пространства;
- устранение шумов;
- нахождение контуров;
- сравнение с эталонным изображением.

Процесс загрузки эталонных изображений можно описать следующим образом:

```
Forfardleft = cv2.imread('D:/ffff/1936-articles-4-1-5.png')
Right = cv2.imread('D:/ffff/new.JPG')
Forfardleft=cv2.resize(Forfardleft,(64,64))
Right = cv2.resize(Right,(64,64))
Forfardleft = cv2.inRange(Forfardleft,(89,91,149),(255,255,255))
Right = cv2.inRange(Right,(89,91,149),(255,255,255))
#cv2.imshow("Forfardleft",Forfardleft)
#cv2.imshow("Right",Right)

FORWARD=cv2.imread('D:/ffff/forvard.jpg')
FORWARD=cv2.resize(FORWARD,(64,64))
FORWARD = cv2.inRange(FORWARD,(89,91,149),(255,255,255))
#cv2.imshow("FORWARD",FORWARD)

Left=cv2.imread('D:/ffff/left new.JPG')
Left=cv2.resize(Left,(64,64))
Left = cv2.inRange(Left,(89,91,149),(255,255,255))
#cv2.imshow("Left",Left)

FORWARD_AND_RIGHT=cv2.imread('D:/ffff/fr new.JPG')
FORWARD_AND_RIGHT=cv2.resize(FORWARD_AND_RIGHT,(64,64))
FORWARD_AND_RIGHT = cv2.inRange(FORWARD_AND_RIGHT,(89,91,149),(255,255,255))
#cv2.imshow("FORWARD AND RIGHT",FORWARD_AND_RIGHT)
```

Разберем процесс на примере знака «Прямо и направо».

Сначала считываем из памяти эталонное изображение знака, например, «Прямо и направо». Это осуществляется директивой

FORWARD_AND_RIGHT=cv2.imread('D:/ffff/fr new.JPG').

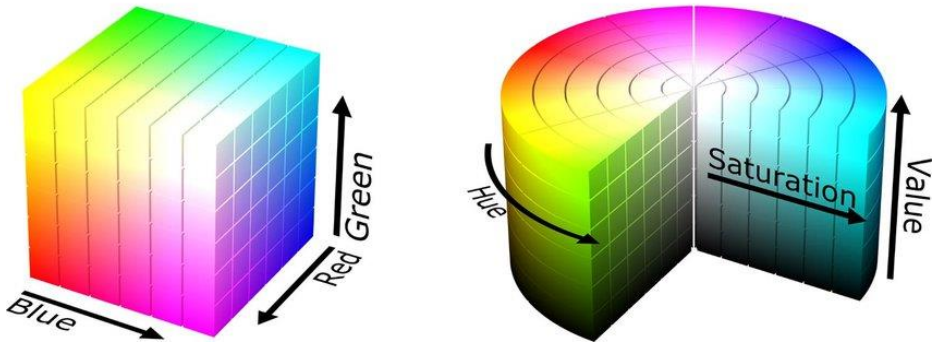
Далее происходит изменение размера эталонного знака до формата 64x64 для уменьшения количества обрабатываемой информации.

FORWARD_AND_RIGHT = cv2.inRange(FORWARD_AND_RIGHT,(89,91,149),(255,255,255))

Для достоверного распознавания знака необходимо использовать наложение цветового фильтра. Данный фильтр с RGB параметрами (89, 91, 149), (255, 255, 255) устанавливает разрешенные цвета. В описываемом случае — это красный и синий (поскольку в некоторых случаях возможно распознавание знака «проезд запрещен»). Также это делается для того, чтобы уменьшить количество обрабатываемой информации.

Все дорожные знаки имеют похожие параметры (форму, цвет). В данной программе в качестве опорного параметра использован цвет.

Для распознаваемых знаков основным цветом являются синий и красный. Для извлечения красного или синего цвета из входного изображения используется цветовое пространство HSV (***Hue, Saturation, Value***). Сначала изображение конвертируется из цветового пространства RGB в HSV (рис. 2).



a) *b)*

Рис. 2. Цветовые модели: a) RGB; b) HSV

Рис. 3. Устранение шумов с изображения

Далее необходимо устранить шумы с изображения с помощью функции

```
hsv=cv2.blur(hsv,(5,5))
```

На следующем шаге производится бинаризация изображения, т.е. деление всех цветов на синие (или красные) и все остальные.

Рис. 4. Бинаризация изображения

Когда синий (или красный) цвет будет обнаружен, можно определить контур области заданного цвета:

(cv2.findContours())

Далее обводим эту синюю или красную область в виртуальный квадрат:

(cv2.rectangle(frame2,(x,y),(x+w,y+h),(0,255,0),2))

Полученное квадратное изображение изменяем до размеров 64x64 пикселей:

(roImg=cv2.resize(roImg,(64,64)))

и пропускаем его через такой же фильтр, что и эталонные изображения:

(roImg=cv2.inRange(roImg,(89,124,73),(255,255,255)))

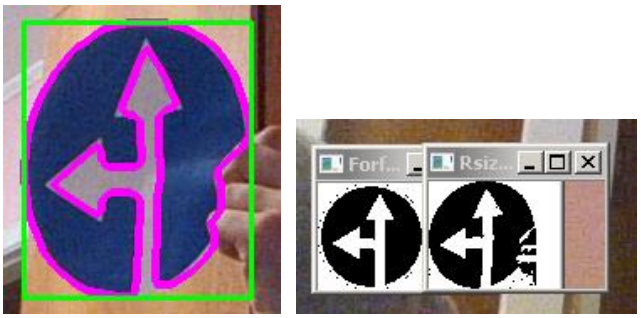


Рис. 5. Определение контура заданного цвета

После фильтрации необходимо вывести количество совпавших пикселей в консоль для отладки:

```
print(FORWARD_AND_RIGHT_val," ^ ", Right_val," ^ ",FORWARD_val," ^
",Forfardleft_val," ^ ",Left_val)
```

Следовательно, поднеся к камере какой-либо знак, один из столбцов в консоли будет иметь наибольшее значение. Зная какой столбец какому знаку принадлежит, можно определить сам знак. Например, при поднесении знака «FORWARD_AND_RIGHT», первый столбец имел значение 2900, а остальные варьировались от 2300 до 2500. Далее, зная примерное количество пикселей при совпадении, можно написать такой код для определения знака:

```
if FORWARD_val>3000:
            b=[20]
        ser.write('b')
            print("FORWARD")
        elif Forfardleft_val>2850:
            b=[20]
            ser.write('b')
            print("FORWARD AND LEFT")
        elif FORWARD_AND_RIGHT_val>2800:
            b=[30]
            ser.write('b')
            print("FORWARD AND RIGHT")
        elif Right_val>2800:
            b=[40]
            ser.write('b')
            print("RIGHT")
        elif Left_val>2750:
```

```
            b=[50]
            ser.write('b')
            print("LEFT")
        else:
            print("nothing")
```

Когда знак определен, отправляем соответствующий сигнал по COM порту к главному микрокомпьютеру, где находится программа управления логикой движения автономного мобильного робота.

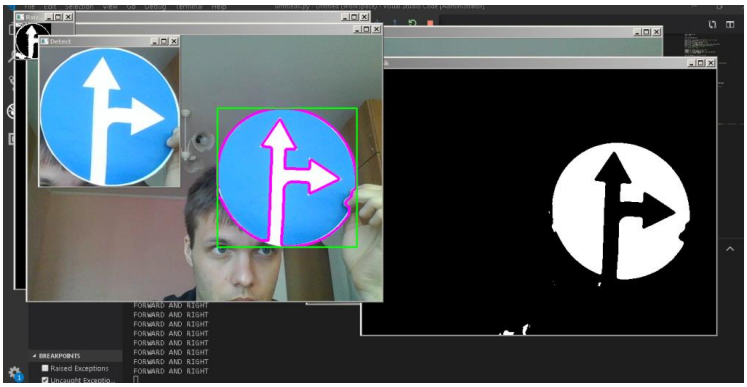


Рис. 6. Пример формирования управляющего воздействия

Метод распознавания формы. Этот метод обнаружения знаков в качестве базового параметра использует не цвет, а форму. Все знаки, которые должны были быть обнаружены, имеют круглую форму.

Метод основан на том, что сначала распознается окружность, которая попала в объектив видеокамеры, затем эта окружность делится на 9 фрагментов, из этих девяти фрагментов выбирается четыре ключевых фрагмента, в которых могут находиться белые части знака. Далее используется функция

get_dominant_color,

которая находит доминирующий цвет в рассматриваемом фрагменте. Например, если белый цвет наблюдался только в верхнем и нижнем фрагментах, то можно утверждать, что это знак «только прямо».

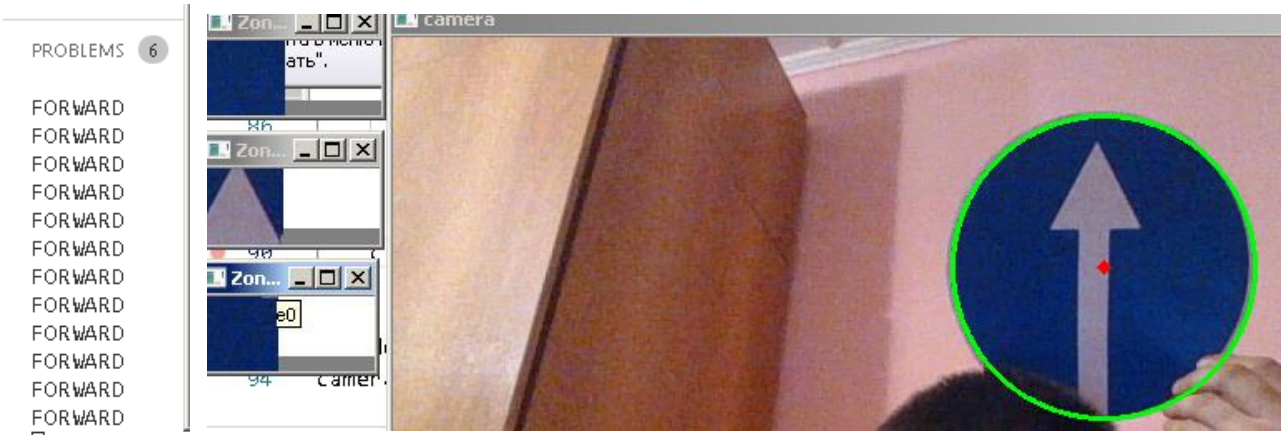


Рис. 7. Пример распознавания формы знака

Заключение. Оба метода показали себя хорошо. Однако недостаток первого метода заключался в том, что, если знак находился на синем или красном фоне, то его было сложно

обнаружить, так как он сливался с фоном. Второй способ не имел такого недостатка, так как в качестве основного параметра была использована форма объекта, а не его цвет.

Библиографический список

1. ПДД РФ, Правила дорожного движения Российской Федерации [Электронный ресурс] / КонсультантПлюс. — Режим доступа : http://www.consultant.ru/document/cons_doc_LAW_2709/824c911000b3626674abf3ad6e38a6f04b8a7428/ (дата обращения :22.05.2019).

2. Visual Studio Code — редактор кода для Linux, OS X и Windows [Электронный ресурс] // Habr. — Режим доступа : https://habr.com/ru/company/ microsoft/blog/262523/ (дата обращения : 15.04.2019).