

Мастеркласс по детектированию по цвету и распознаванию дорожных знаков на изображении используя методы компьютерного зрения.

Цель проведения:

Знакомство с библиотекой компьютерного зрения OpenCV, методами и алгоритмами, включенными в библиотеку OpenCV.

Время проведения: 45 - 60 минут

Оборудование:

- ПК, веб - камера;
- Python 2.7 (numpy, scipy), OpenCV 2.4;

Давайте поговорим о том, какие функции должны выполнять беспилотные автомобили.

Как вы считаете что должен уметь беспилотный автомобиль?

Одной из самых первых функций беспилотных автомобилей является распознавание дорожных знаков. Сама функция распознавания дорожных знаков изначально была разработана для систем помощи водителю и представляла собой программу, способную определять дорожный знак по ходу движения автомобиля и оповещать водителя различными способами о назначении обнаруженного знака. В беспилотных транспортных средствах данная функция оповещает уже не водителя а сам автомобиль. При этом “мозг” автомобиля принимает решение, какие действия необходимо выполнить при обнаружении того или иного знака.

Под мозгом мы подразумеваем что?

Набор программных и аппаратных средств для анализа дорожной ситуации и различных внешних воздействий, принятия решений и выполнения действий на основании принятых решений.

Мы с вами будем говорить про программные средства. Наиболее распространенными технологиями, используемыми в настоящее время в беспилотных автомобилях являются компьютерное зрение и нейронные сети. В особенности сверточные нейронные сети, но об этом позже.

Компьютерное зрение применяется сейчас повсеместно и беспилотные автомобили не исключение.

Что же такое компьютерное зрение? Как вы считаете?

Компьютерное зрение по сути своей - это получение нужной информации из изображения. Обработка изображения таким образом, чтобы получить из него какие-либо данные. Примером таких данных может быть информация об объекте, как например форма или цвет, расположение объекта и расстояние объекта от камеры.

Как например найти дорожный знак на изображении?

Например найти определенный цвет знака, красный или синий, найти контуры для последующего определения расположения на изображении, также зная размер дорожного знака можно экспериментально вычислить расстояние до него.

Давайте попробуем с вами теперь найти знак на изображении.

Существует множество базовых операций обработки изображений, которые помогут нам выделить нужную информацию на изображении. Это фильтрация, геометрические преобразования, преобразование цветовых пространств и т. д.

В качестве основного параметра детектирования возьмем цвет объекта. Это самый распространенный способ выделения объекта на изображении. Однако этот способ имеет и свои недостатки, например, влияние освещенности на цвет объектов или множество других объектов с теми же характеристиками цвета.

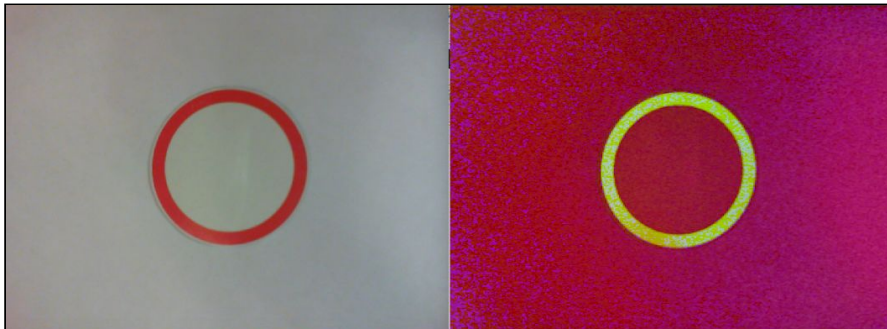
Эти недостатки можно устранить (сделать их влияние незначительным) с помощью различных преобразований изображения (фильтров, накладываемых на изображение).

У каждого из вас есть инструкция об операциях, используемых для выделения объекта по цвету.

Инструкция:

Рассмотрим функции преобразования изображения, необходимые для работы программы распознавания изображений.

cvtColor - конвертирует изображение из одного цветового пространства в другое. Функция предназначена для преобразования цветового пространства RGB в цветовое пространство HSV.



Такое преобразование используется потому, что в пространстве HSV цвет менее зависим от яркости, однако при крайних значениях яркости в канале тона появляется много шума:

```
hsv = cv.cvtColor(frame, cv.COLOR_BGR2HSV)
```

blur – размытие изображения. В программе функция позволяет уменьшить шум на изображении.

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

inRange - позволяет выделить маску области. По сути функция проверяет, находятся ли элементы массива (пиксели изображения) в промежутке между заданными значениями нижней и верхней границы. Значения верхней и нижней границы можно получить экспериментальным путем.

Запустите программу, содержащуюся в файле hsvSettings.py и с помощью ползунков подберите значения нижней и верхней границы цвета. Функция осуществляет бинаризацию изображения. После преобразований на изображении остаются некоторые пятна (шумы), которые не имеют отношения к детектируемому изображению и способны помешать точно определить его:

```
lower = np.array([56, 91, 149])
```

```
upper = np.array([255, 255, 255])  
thresh = cv.inRange(hsv, lower, upper)
```

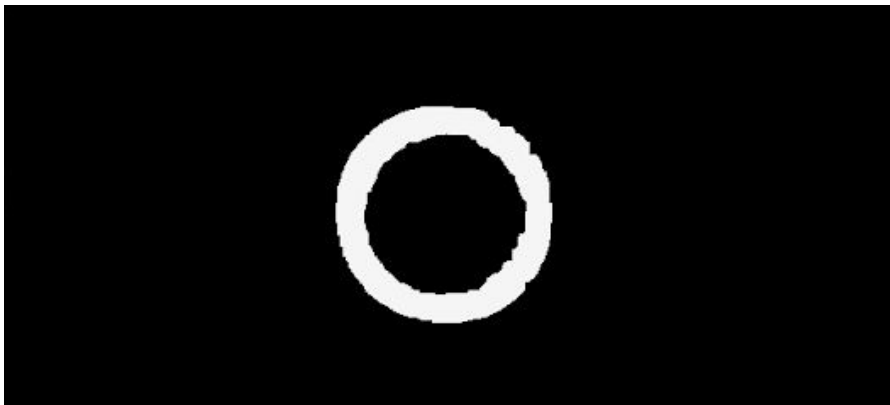
erode - размывание(операция сужения).

dilate - растягивание(операция расширения).

Эрозия убирает белые пиксели с изображения, удаляя мелкие пятна, а дилатация не позволяет крупным областям уменьшиться. Удаленные во время размытия мелкие пятна во время растяжения не появятся снова.

```
thresh = cv.erode(thresh, None, iterations=2)  
thresh = cv.dilate(thresh, None, iterations=6)
```

В результате применения приведенных выше функций получаем бинаризованное изображение с выделенным на нем объектом (см. рис. 44).



Извлечение объекта из начального изображения сводится к выделению части изображения, в которой заключен объект бинаризованного изображения с последующим извлечением этой же части из начального изображения. Для выполнения этой операции следует найти контуры объекта на бинаризованном изображении.

findContours – поиск контуров на изображении. При условии, что бинаризация на первом этапе была выполнена корректно, функция находит контуры объектов на изображении.

Тот факт, что знаки в большинстве своем имеют строго определенную форму (квадрат, треугольник, круг, прямоугольник), дает нам возможность заключить знак в прямоугольную область с равными сторонами.

```
contours, hierarchy = cv.findContours(thresh.copy(), cv.RETR_TREE, cv.CHAIN_APPROX_SIMPLE)
```

Также необходимо выполнить сортировку всех найденных объектов и определить самый большой из них. Сортировка выполняется внутри цикла.

```
for cnt in contours:
```

```
    c = sorted(contours, key=cv.contourArea, reverse=True)[0]
```

minAreaRect – ограничивающий прямоугольник (box). Прямоугольная область, которая ограничивает полученный контур объекта на изображении. После выполнения функции `minAreaRect`, появляется возможность получить координаты четырех точек (углов) ограничивающего прямоугольника.

```
rect = cv.minAreaRect(c)
```

BoxPoints – получение координат четырех точек (углов) ограничивающего прямоугольника.

```
box = np.int0(cv.BoxPoints(rect))
```

Отобразим полученный многоугольник на основном изображении.

```
cv.drawContours(frame, [box], -1, (0, 255, 0), 3)
```

ДОПОЛНИТЕЛЬНО

Чтобы извлечь нужный фрагмент изображения с детектируемым объектом, необходимо получить координаты объекта по двум точкам.

```
y1 = int(box[0][1])
```

```
x2 = int(box[1][0])
```

```
y2 = int(box[1][1])
```

```
x3 = int(box[2][0])
```

```
roilmg = frame[y2:y1, x2:x3]
```

Чтобы отобразить вырезанный фрагмент изображения, необходимо выполнить команду

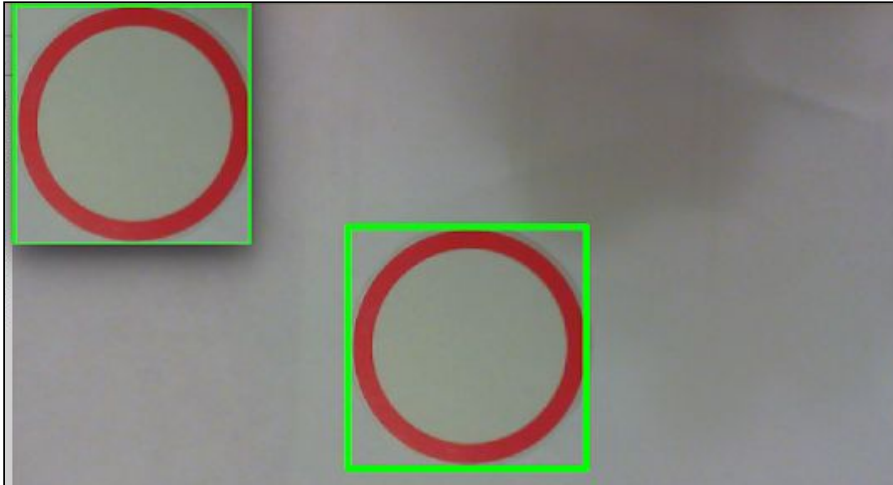
```
cv.imshow('roilmg',roilmg)
```

Однако, если полученный объект исчезнет из кадра, программа завершит работу с ошибкой. Чтобы этого не произошло, необходимо проверить наличие фрагмента.

```
if roilmg.any():
```

```
cv.imshow('roimg',roimg)
```

Дальнейшие операции необходимо выполнять внутри проверки. При копировании из исходного изображения области с координатами четырех точек (углов) прямоугольника, получаем новое изображение, которое соответствует изображению детектируемого объекта.



Чтобы сравнить полученное изображение с некоторым шаблонным файлом, необходимо открыть шаблонное изображение, привести оба изображения к одному размеру, осуществить бинаризацию изображений и попиксельно сравнить оба изображения.

```
noDrive = cv.imread("noDrive.png")
resizedRoi = cv.resize(roimg, (100, 100))
noDrive=cv.resize(noDrive,(100, 100))

xresizedRoi=cv.inRange(resizedRoi, lower, upper)
xnoDrive=cv.inRange(noDrive, lower, upper)
identity_percent=0
for i in range(100):
    for j in range(100):
        if (xresizedRoi[i][j]==xnoDrive[i][j]):
            identity_percent=identity_percent+1
print identity_percent
```

Чем выше полученное значение `identity_percent`, тем больше совпадение шаблона и детектируемого объекта на изображении.