Повторитель дорожных знаков голосом

Разработал ученик 10Е класса ГБОУ школы №2045 имени Дмитрия Разумовского Виктор корсак Александрович , Алексеев Илья Романович

Руководитель:Ступин Артем Сергеевич

Оглавление

Описание идеи проекта

Сейчас на дорогах происходит множество аварий и ДТП из за невнимательности людей на дорожные знаки по типу: Главная дорога или ограничение скорости .Наш проект поможет водителям на дороге и предотвратит множество аварий и ДТП и нарушений правил дорожного движения

Цель и задачи проекта

Узнать из за незаметности каких знаков случаются аварии и узнать, как научить программу распознавать знаки и сообщать о них .

Задачи :

-изучить принцип работы камеры и звукового устройства

-выяснить как современные камеры распознают знаки

Анализ рынка

на современном рынке представлены разные навигаторы но в основном они не показывают множество знаков. И регистраторы которые в основном показывают только ограничение скорости

Наш проект будет же не показывать ограничение скорости и другие знаки , а их озвучивать тем самым не отвлекая водителя во время езды

SWOT-Анализ

преимущества - универсальность (можно встроить в любой автомобиль. Недостатки - при установке в автомобиль нужно перенастраивать цвета на камере.

Методика выполнения

Для реализации данного алгоритма был выбран язык программирования Python3. Область применения Python очень обширна. Создавались стандартные библиотеки для поддержки современных технологий в том числе и библиотеки для работы со звуком и обработкой видео. Такими библиотеками стали OpenCV и Pygame.mixer.

Идея «Распознавания знаков» основана на библиотеке OpenCV.

Определение нахождения, цвета и типа знаков — все это возможно с помощью библиотеки OpenCV.

```
maskb = cv2.inRange(img, lowerb: (0, 0, 0), upperb: (255, 255, 0))
maskr = cv2.inRange(img, lowerb: (0, 26, 70), upperb: (106, 101, 226))
masky = cv2.inRange(img, lowerb: (0, 0, 135), upperb: (130, 180, 202))
maskbl = cv2.inRange(img, lowerb: (0,0,0), upperb: (87,97,82))
```

Сначала создается бинарная маска по которой будет отбираться цвет знаков

. В параметрах указано изображение по которому будет создана маска(в нашем случае по изображению из веб-камеры),

далее нижняя и верхняя граница цвета.

```
contoursr, hierarchyr = cv2.findContours(maskr, cv2.RETR_TREE, cv2.CHAIN_APPROX_NONE)
contoursb, hierarchyb = cv2.findContours(maskb, cv2.RETR_TREE, cv2.CHAIN_APPROX_NONE)
contoursy, hierarchyy = cv2.findContours(masky, cv2.RETR_TREE, cv2.CHAIN_APPROX_NONE)
contoursbl, hierarchybl = cv2.findContours(maskbl, cv2.RETR_TREE, cv2.CHAIN_APPROX_NONE)
cv2.drawContours(img, contoursy, -1, color: (255, 0, 255), thickness: 3)
cv2.drawContours(img, contoursb, -1, color: (0, 0, 255), thickness: 3)
cv2.drawContours(img, contoursr, -1, color: (255, 0, 0), thickness: 3)
cv2.drawContours(img, contoursr, -1, color: (255, 255, 0), thickness: 3)
```

Далее программа ищет контуры цветов и обрисовывает их.

Следующим шагом программа проверяет наличие контуров. Если они есть,
тогда компьютер ищет самый большой конур и обрисовывает его в квадрат.
Потом по этому квадрату определяется положение знака и в новую переменную записывается картинка со считавшимся знаком.
Далее полученное изображение уменьшается и бинаризируется.
На данном фрагменте представлена обработка желтого цвета.

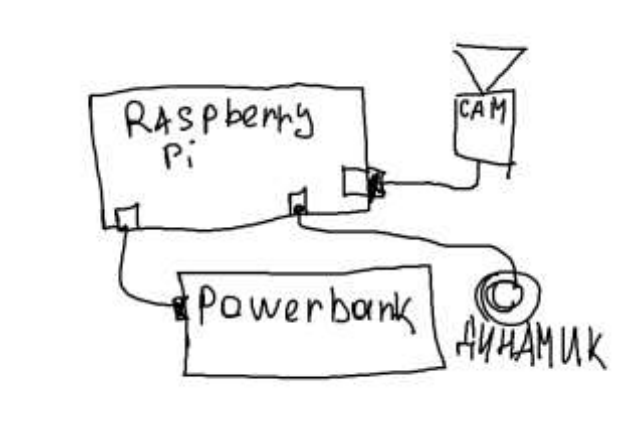
```
if len(contoursy) != 0:
    contoursy = sorted(contoursy, key=cv2.contourArea, reverse=True)
    cv2.drawContours(frame, contoursy, contourIdx: 0, color: (255, 0, 255), thickness: 3)
    #cv2.imshow("Contors", frame)

    (xy, yy, wy, hy) = cv2.boundingRect(contoursy[0])
    cv2.rectangle(frame, (xy, yy), (xy+wy, yy+hy), (0, 255, 0), 2)
    #cv2.imshow("Rect", frame)

    roImgy = frameCopy[yy:yy+hy, xy:xy+wy]
    #cv2.imshow("DetectY", roImgy)
    roImgy = cv2.resize(roImgy, dsize: (64,64))
    roImgy = cv2.inRange(roImgy, lowerb: (0, 0, 135), upperb: (130, 180, 202))
    #cv2.imshow("ResizedRoiy", roImgy)
```

Следующим шагом программа попиксельно сравнивает полученное обработанное изображение с веб-камеры и сравнивает его с шаблонами знаков.
Если большинство пикселей совпадает с шаблоном, то программа сообщает про увиденный знак.

```
for i in range(64):
    for j in range(64):
        if len(contoursy) != 0:
            if roImgy[i][j] == MainWay[i][j]:
                mainWay += 1
        if len(contoursr) != 0 and len(contoursbl) != 0:
            if roImgblr[i][j] == DangerousBandsLeft[i][j]:
                dangerousBandsLeft += 1
            if roImgblr[i][j] == DangerousBandsRight[i][j]:
                dangerousBandsRight += 1
            if roImgblr[i][j] == DangerousBandLeft[i][j]:
                dangerousBandLeft += 1
            if roImgblr[i][j] == DangerousBandRight[i][j]:
                dangerousBandRight += 1
```



На данном изображении представлена схема проекта, состоящая из Raspberry pi, блока питания (powerbank), динамика и камеры (cam).
Идея «Распознавания знаков» основана на библиотеке OpenCV.
Определение нахождения, цвета и типа знаков — все это возможно с помощью библиотеки OpenCV.

СРОКИ РЕАЛИЗАЦИИ

+ 1-7 Декабря - Анализ знаков и выбор одного из них
+ 8 Декабря - 10 Января - создание кода проекта
+ 10 - 20 января -бета тест
+ 20 - 14 февраля - исправления ошибок

Смета расходов

+ Камера 2000-2500 руб.
+ Raspberry pi 5000-7000 руб.
+ Powerbank 2000 руб.
+ Динамик 2000 руб.

Описание результатов

На данный момент программа с помощью веб камеры компьютера распознает знаки на расстоянии 15 метров и выводит их в виде голосового сообщения

Перспективы и выводы

Проект должен уменьшить количество дтп на дороге. В перспективе добавление новых видов знаков и улучшение методов распознавания