МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра Систем автоматического управления

ОТЧЕТ

по производственной практике

Тема: Разработка системы распознавания сигналов светофора

Студент гр. 9491 _____ Кустов Д.И.

Руководитель _____ Кузьмина Т.О.

Санкт-Петербург
2023

ЗАДАНИЕ

НА ПРОИЗВОДСТВЕННУЮ ПРАКТИКУ

Студент Кустов Д.И.

Группа 9491

Тема практики: Разработка системы распознавания сигналов светофора

Задание на практику:

- Изучение существующих решений по данной тематике;
- Выбор необходимых инструментов и библиотек;
- Создание базовой версии программы;

Сроки прохождения практики: 09.01.2023 –04.02.2023

Дата сдачи отчета: 02.02.2023

Дата защиты отчета: 02.02.2023

Студент _____ Кустов Д.И.

Руководитель _____ Федоркова А.О.

Руководитель от кафедры _____ Кузьмина Т.О.

АННОТАЦИЯ

Работа посвящена обзору методов распознавания сигналов светофора с использованием библиотеки технического зрения OpenCV. Основное содержание охватывает использование функций и методов OpenCV для обнаружения классификации сигналов светофора в видеопотоке данных. Основное внимание уделяется базовым методам распознавания цветов сигнала светофора в пространстве HSV и практической реализации по разработке системы распознавания сигналов светофора.

SUMMARY

The work is devoted to an overview of methods for recognizing traffic signals using the OpenCV vision library. The main content covers the use of OpenCV functions and methods to detect the classification of traffic signals in a video data stream. The main attention is paid to the basic methods of traffic light color recognition in the HSV space and practical implementation for the development of a traffic light recognition system.

СОДЕРЖАНИЕ

	Введение	5
1.	Основные теоретические сведения	6
1.1.	Что такое техническое зрение и какие задачи оно решает	6
1.2.	Существующие методы распознавания	8
2.	Практическая реализация	10
2.1.	Разработка программы для распознавания сигналов светофора	10
2.2.	Результат работы программы для распознавания сигналов светофора	13
	Заключение	15
	Список использованных источников	16
	Приложение А. Код программы	17

ВВЕДЕНИЕ

Использование технологий технического зрения становится все более важным в последние годы, поскольку растет спрос на автономные транспортные средства и интеллектуальные транспортные системы. В этом контексте, распознавание сигналов светофора является важнейшим аспектом обеспечения безопасности дорожного движения и эффективной транспортировки. Здесь на помощь приходит библиотека технического зрения OpenCV. OpenCV предоставляет широкий спектр инструментов для обработки изображений и видео, что делает ее лучшей библиотекой для разработки системы распознавания сигналов дорожного движения в реальном времени.

В данной работе, основное внимание будет уделено использованию алгоритмов технического зрения для анализа видеопотока данных и определения наличия сигналов светофора. Работа будет включать в себя использование различных методов, таких как, сегментация изображений, работа в пространстве HSV и применение цветовых фильтров, чтобы научить систему точно распознавать различные типы сигналов светофора.

Таким образом, после того, как система будет разработана, ее можно интегрировать в автономные транспортные средства, роботов или другие интеллектуальные транспортные системы, предоставляя в режиме реального времени информацию о наличии сигналов светофора. Это поможет повысить безопасность дорожного движения, обеспечив правильную реакцию транспортных средств на сигналы светофора, что снизит риск аварий и чрезвычайных ситуаций.

1. ОСНОВНЫЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

1.1ЧТО ТАКОЕ ТЕХНИЧЕСКОЕ ЗРЕНИЕ И КАКИЕ ЗАДАЧИ ОНО РЕШАЕТ

Техническое зрение – это технология создания таких автоматизированных систем, которые способны выполнять такие задачи, как:

- наблюдение за производственной линией;
- отслеживание ошибок и неисправностей;
- сортировка продукций и многое другое

Данная область объединяет такие сферы, как: робототехника, компьютерные сети, цифровые технологии. Так же, техническое зрение является частью искусственного интеллекта.

Основная цель технического зрения – это определение, исследование и обнаружение различных объектов. Данная технология получает необходимую информацию из изображения или серии изображений (видеопотока). Задачи могут быть самыми разными, такими как:

- Распознавание лиц
- Детектирование границ, разметок
- Определение движения объектов (Motion Tracking)
- Задачи распознавания объектов
- Распознавание дорожных знаков
- Распознавание сигналов светофора
- Задачи реконструкции сцены
- Восстановление изображений

Техническое зрение все больше развивается, и находит свое применение в самых разных сферах, касающихся обработки изображений и распознавания образов в бытовых, промышленных и военных сферах.

Для решения задач, связанных с техническим зрением, используется библиотека OpenCV.

OpenCV (Open Source Computer Vision) – это библиотека программных функций, с открытым исходным кодом, которые используются для технического зрения. Данная библиотека написана на языке C++, но интерпретирована во многих других популярных языках. Она предоставляет базовые средства для решения задач технического зрения любого уровни сложности, и одновременно дает простор для дополнения получившихся решений разработчиком. Главным достоинством OpenCV является то, что данная библиотека является кроссплатформенной и бесплатной для использования по лицензии в различных целях (Учебных и коммерческих).

В нашей работе, мы будем рассматривать использование OpenCV при программировании на языке Python в среде разработки PyCharm.

Библиотека OpenCV применяется:

- **В робототехнике —** для ориентирования робота в пространстве, распознавания объектов и взаимодействия с ними;
- **Медицинских технологиях —** для создания точных методов диагностики, например 3D-визуализации органа при МРТ;
- **Промышленных технологиях —** для автоматизированного контроля качества, считывания этикеток, сортировки продуктов и пр.;
- **Безопасности —** для создания «умных» камер видеонаблюдения, которые реагируют на подозрительные действия, для считывания и распознавания биометрии;
- **На транспорте —** для разработки автопилотов.

1.2 СУЩЕСТВУЮЩИЕ МЕТОДЫ РАСПОЗНАВАНИЯ

Для выполнения поставленной задачи, был выбран метод распознавания с использованием цветового пространства HSV. Плюсом данного метода является то, что данное цветовое пространство разлагает цвета по яркости, в отличии от привычного нами RGB пространства (RGB пространство наоборот – складывает цвета).

Цветовое пространство – это определенная организация цветов, которая позволяет нам последовательно представлять и производить цвета. Оно определяет одно, двух, трёх, или четырехмерное пространство, чьи измерения, или компоненты**,** представляют собой значения интенсивности цвета. Цветной компонент также называют цветным каналом.

К примеру, **RGB-пространство** - это трехмерное цветовое пространство, в которой красный, зеленый и синий цвета складываются вместе различными способами для воспроизведения широкого спектра цветов. Визуально эти пространства часто представляют различными объемными фигурами - кубами, конусами или многогранниками. На рис.1.1 представлено цветовое пространство RGB:

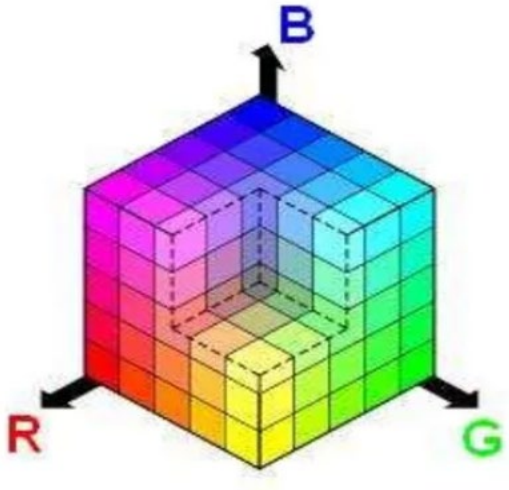

Рисунок 1.1 – цветовое пространство RGB

Цветовое пространство **HSV** разлагает яркость цвета и широко используется в алгоритмах улучшения изображения. В отличие от RGB, HSV пространство представляет собой фигуру в форме конуса. Данная цветовая схема определяется тремя компонентами:

- Hue - цветовой тон;
- Saturation - насыщенность;
- Value - яркость;

Hue - цветовой тон. Оттенок представляет цвет. Определяется, какой «чистый» цвет исследуется. К примеру, все тени и тона цвета «красный» будут иметь одинаковый оттенок. Здесь оттенок варьируется в пределах 0—360°, однако иногда приводится диапазон 0-100 или 0-1.

Saturation - насыщенность. Насыщенность указывает диапазон серого в цветовом пространстве. Варьируется в диапазоне 0 – 100 % . Иногда значение вычисляется от 0 до 1. Когда значение равнение равно 0, цвет становится серым, когда равно 1 – цвет является основным цветом.

Value – яркость цвета, которая зависит от насыщенности цвета. Варьируется в диапазоне 0 – 100 % или 0 – 1. Когда значение равно 0 – цветовое пространство будет полностью черным. С увеличением значения, цветовое пространство становится ярче и отображает различные цвета. На рис.1.2 представлено цветовое пространство HSV:

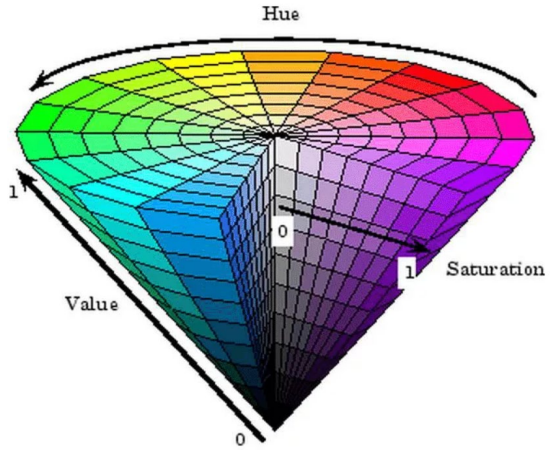

Рисунок 1.2 – цветовое пространство HSV

2. ПРАКТИЧЕСКАЯ РЕАЛИЗАЦИЯ

2.1 РАЗРАБОТКА ПРОГРАММЫ ДЛЯ РАСПОЗНАВАНИЯ СИГНАЛОВ СВЕТОФОРА

Сигналы светофоров являются важными элементами регулирования движения на дорожном полотне. Их классификация по цвету сигнала помогает автономным автомобилям принимать решения об остановке или движении. Все светофоры похожи по форме и внешнему виду (рис. 2.1). Для распознавания используется детектор, основанный на гистограмме направленных градиентов и методе опорных векторов. Для обучения детектора необходима выборка изображений с детектируемым объектом. Эти изображения должны быть размечены, чтобы показать координаты искомых областей. Такой набор данных позволит создать массив направлений градиентов для каждого элемента выборки и обучить детектор.

Рисунок 2.1 – внешний вид светофора

Главной задачей при разработке программы являлось определение в видеопотоке переключение кадра сигнала светофора с одного цвета на другой.

Для разработки программы был использован следующий алгоритм:

- Находим все «красные» участки в кадре, кладем их все в массив, связанный с кадром.
- Находим в текущем кадре все пропавшие «красные» участки – добавляем в массив возможных красных сигналов светофора текущего кадра
- Так же находим все «желтые» и «зеленые» участки в кадре – кладем в массив
- Проверяем все возможные сигналы, на то, что они близки по форме к кругу
- Делаем проверку на относительные размеры и на взаимное расположение
- При нахождение удовлетворяющей пары – нашли переключение светофора

Как говорилось ранее, для детектирования сигналов светофора, было использовано пространство HSV. Это пространство использовалось для того, чтобы распознавать все оттенки того или иного цвета. Например, красный сигнал светофора не является «чистым», а может иметь разный оттенок – от оранжевого до фиолетового. Плюс ко всему, светофоры могут быть основаны на различных технологиях, от ламп накаливания до светодиодов, имея при этом разный оттенок. Для более детального обзора, приведем пространство HSV в виде окружности (см. рис.2.2).

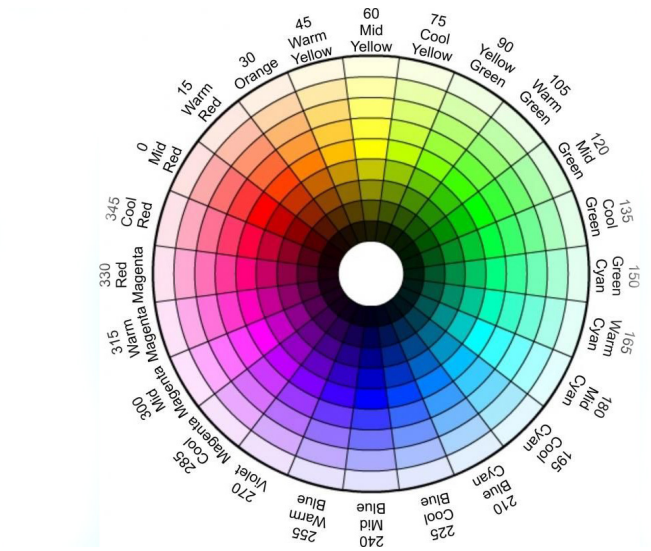

Рисунок 2.2 – цветовое пространство HSV

Выбор области того или иного цвета на самом деле очень много значит, так как увеличение числа оттенков позволит охватить большее число различных вариаций светофоров, регистраторов и погодных условий. Но чрезмерное увеличение данной области приведет к случайным срабатываниям, что сведет ценность всей системы на нет.

Диапазон каждого цвета для нашего задания и общая маска приведен в коде:

```
lower_red = np.array([0, 137, 249], dtype="uint8")
upper_red = np.array([15, 255, 255], dtype="uint8")
lower_yellow = np.array([17, 165, 130], dtype="uint8")
upper_yellow = np.array([101, 255, 255], dtype="uint8")
lower_green = np.array([40, 85, 180], dtype="uint8")
upper_green = np.array([91, 255, 255], dtype="uint8")

mask_red = cv2.inRange(frame_hsv, lower_red, upper_red)
mask_yellow = cv2.inRange(frame_hsv, lower_yellow, upper_yellow)
mask_green = cv2.inRange(frame_hsv, lower_green, upper_green)

mask_full = mask_red + mask_yellow + mask_green
mask_full = cv2.bitwise_and(frame, frame, mask=mask_full)
cv2.imshow("mask_hsv", mask_full)
```

Результат работы маски приведен на рис.2.3:

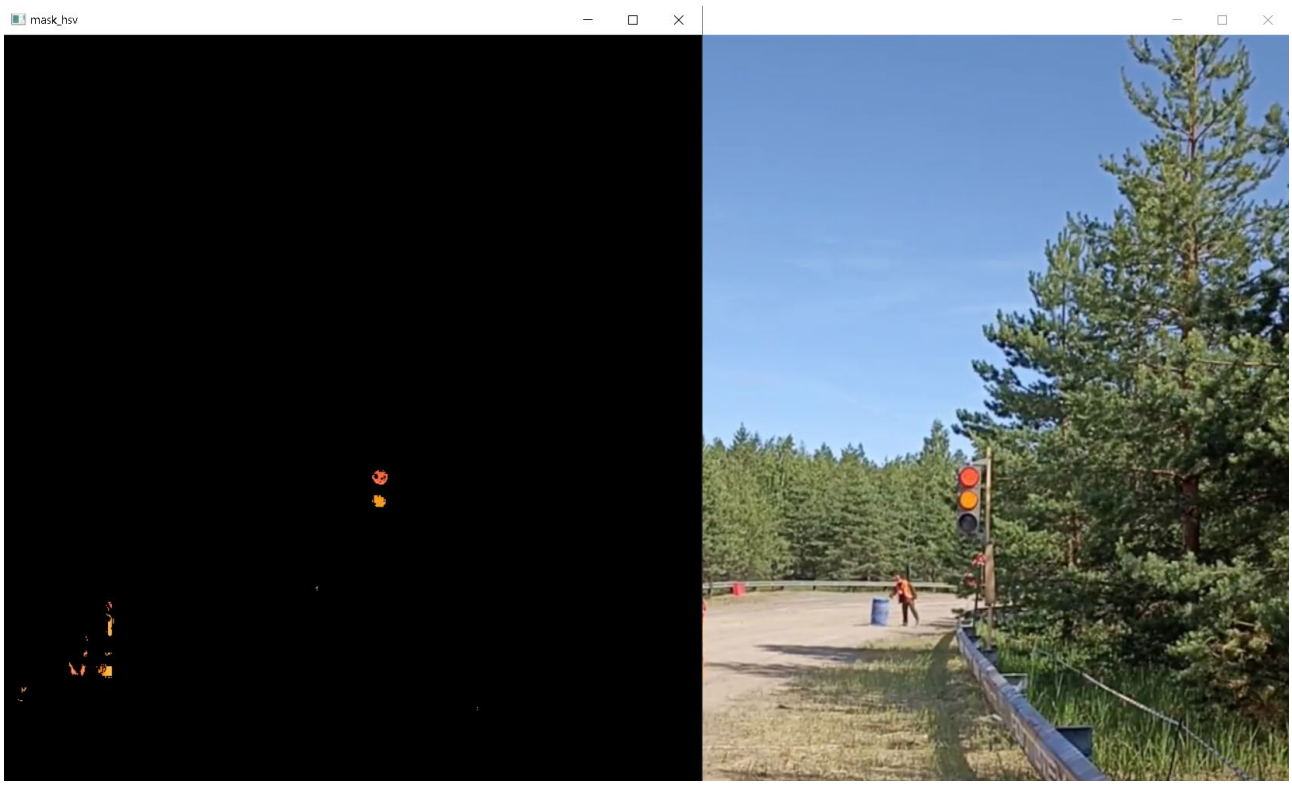

Рисунок 2.3 – маска всех сигналов светофора в цветовом пространстве HSV

Как мы видим, маска отлично срабатывает на все сигналы светофора, исключая большое количество шумовых сигналов, на которые детектор может ложно сработать.

2.2 РЕЗУЛЬТАТ РАБОТЫ ПРОГРАММЫ РАСПОЗНАВАНИЯ СИГНАЛОВ СВЕТОФОРА

В данном подразделе приведен результат работы программы, который распознает в поставленной задаче сигналы светофора. Результат работы для красного, желтого и зеленого цвета приведен на рисунках 2.4, 2.5 и 2.6.

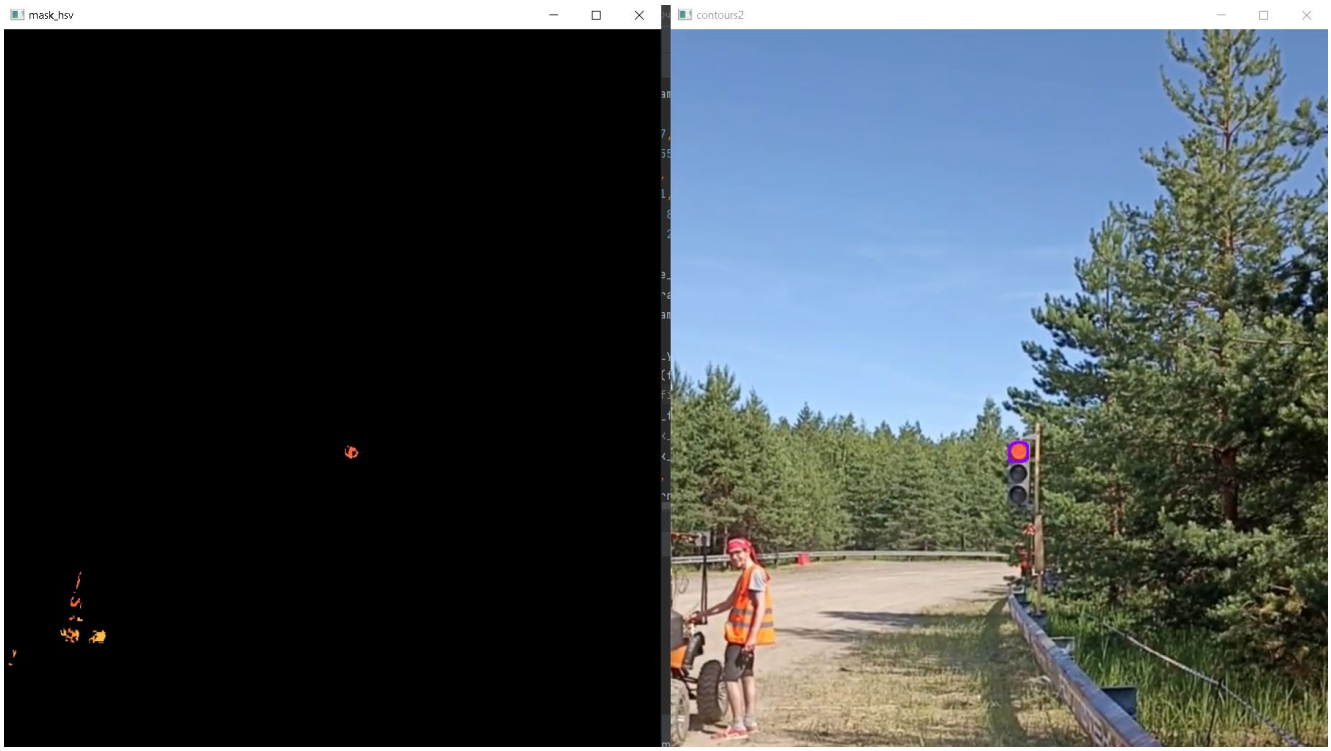

Рисунок 2.4 – детектирование красного сигнала светофора

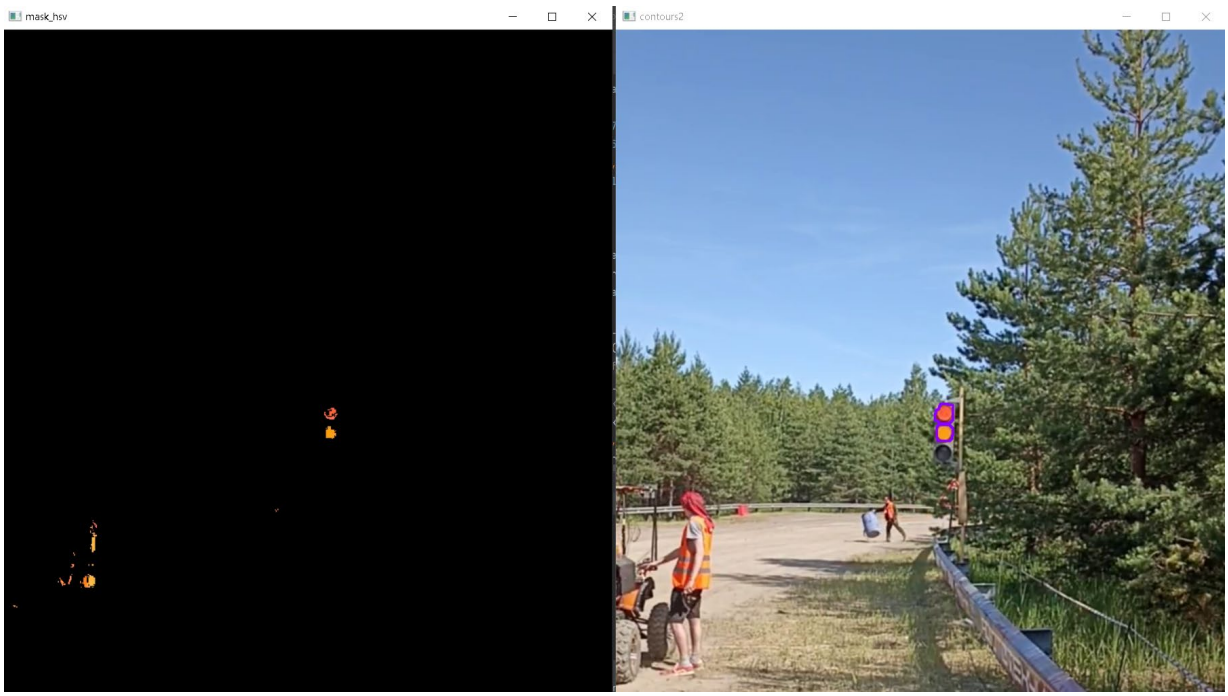

Рисунок 2.5 – детектирование желтого сигнала светофора

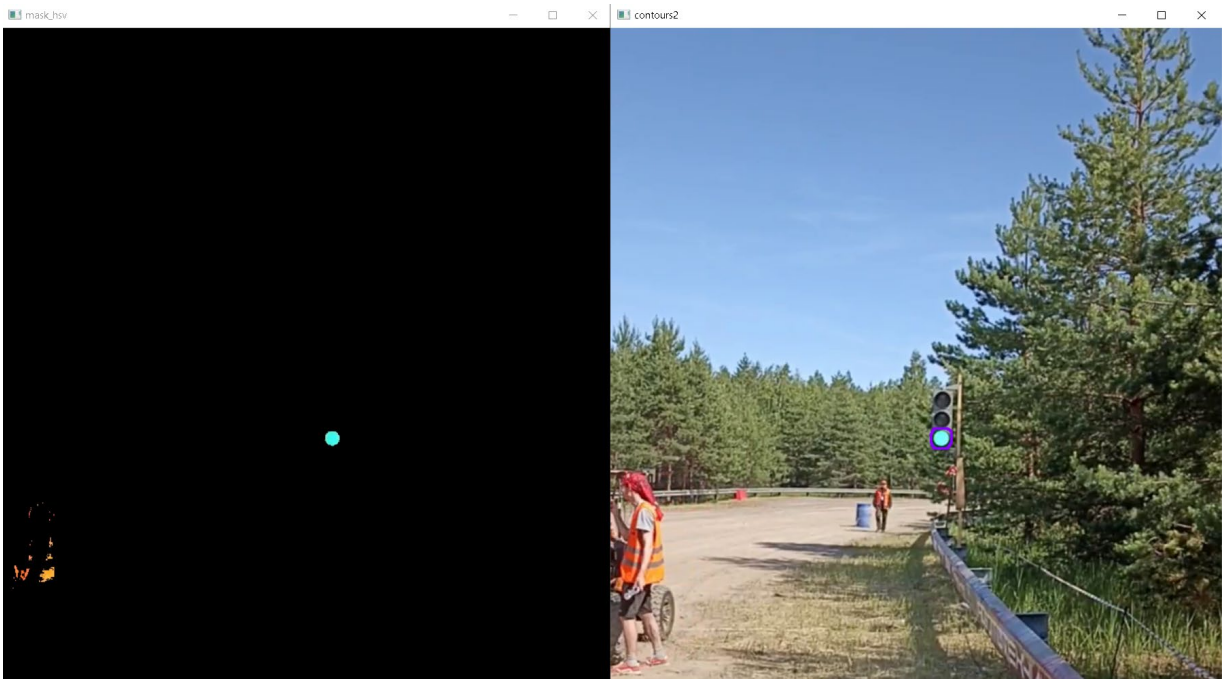

Рисунок 2.6 – детектирование зеленого сигнала светофора

Таким образом, можно сделать вывод о том, что написанная нами программа детектирования сигналов светофора прекрасно справляется с поставленной задачей. Шумы практически отсутствуют, сигнал четко находится по форме и цвету. В Приложении А приведен полный код программы.

ЗАКЛЮЧЕНИЕ

В результате выполнения поставленной задачи, была разработана первая версия программы распознавания сигналов светофора с использованием библиотеки технического зрения OpenCV, которая показала хорошие результаты. Система смогла точно обнаруживать и классифицировать сигналы светофора на видеозаписи, доказывая эффективность методов технического зрения при решении различных типов задач. Использование библиотеки OpenCV позволило найти эффективное решение (работа в пространстве HSV), что сделало его приемлемым для практической реализации. Конечные результаты работы программы демонстрируют потенциал для дальнейшего развития и расширения системы, что приведет к повышению безопасности дорожного движения и созданию более интеллектуальной транспортной системы.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1.Документация по работе с OpenCV [Электронный ресурс] – URL: https://docs.opencv.org/4.x/df/d9d/tutorial_py_colorspaces.html (дата обращения: 22.01.2023)

2.Описание обнаружения светофора с использованием цветовых пространств [Электронный ресурс] – URL: https://www.ijcaonline.org/archives/volume173/number4/verma-2017-ijca-915291.pdf (дата обращения: 22.01.2023)

3.Классификатор светофора [Электронный ресурс] – URL: https://alyxion.github.io/Udacity_IntroToSelfDrivingCarsNd/8_2_Project_5_Traffic_Light_Classifier/Traffic_Light_Classifier.html (дата обращения: 22.01.2023)

ПРИЛОЖЕНИЕ А

КОД ПРОГРАММЫ

```
import numpy as np
import cv2
cap = cv2.VideoCapture('3.mp4')
if not cap.isOpened():
    print("Error opening video")

while (cap.isOpened()):
    ret, frame = cap.read()
    cv2.imshow("video_frame", frame)
    kernel = np.ones((3, 3), np.uint8)
    frame_hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)

    lower_red = np.array([0, 137, 249], dtype="uint8")
    upper_red = np.array([15, 255, 255], dtype="uint8")
    lower_yellow = np.array([17, 165, 130], dtype="uint8")
    upper_yellow = np.array([101, 255, 255], dtype="uint8")
    lower_green = np.array([40, 85, 180], dtype="uint8")
    upper_green = np.array([91, 255, 255], dtype="uint8")

    mask_red = cv2.inRange(frame_hsv, lower_red, upper_red)
    mask_yellow = cv2.inRange(frame_hsv, lower_yellow, upper_yellow)
    mask_green = cv2.inRange(frame_hsv, lower_green, upper_green)

    mask_full = mask_red + mask_yellow + mask_green
    mask_full = cv2.bitwise_and(frame, frame, mask=mask_full)
    cv2.imshow("mask_hsv", mask_full)

    pic3 = cv2.GaussianBlur(mask_full, (13, 13), 4)
    edges = cv2.Canny(pic3, 130, 85, 7)
    pic1 = cv2.dilate(edges, kernel, iterations=4)
    cv2.imshow("mask_dilation", pic1)
    a = frame.shape[0]
    b=frame.shape[1]
    print([a,b])
    contours, hierarcy = cv2.findContours(pic1, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
    for i in range(len(contours)):
        (x, y), radius = cv2.minEnclosingCircle(contours[i])
        if x < 400 and y < 500:
            if cv2.contourArea(contours[i]) > 100:
                con = cv2.drawContours(frame, contours, i, [255, 0, 139],
2)
                stop = 1
                cv2.imshow('contours2', con)
            else:
                stop = 0
                cv2.imshow('contours2', frame)
            print(stop)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
cap.release()
cv2.destroyAllWindows()
```