Международный журнал прикладных наук и технологий "Integral"

Научная статья

Original article

УДК 004.85

РАСПОЗНАВАНИЕ ДОРОЖНОЙ РАЗМЕТКИ С ПОМОЩЬЮ КОМПЬЮТЕРНОГО ЗРЕНИЯ

RECOGNITION OF ROAD MARKINGS USING COMPUTER VISION

Стахеева Алина Алексеевна, магистрант, Северный (Арктический) федеральный университет им. М. В. Ломоносова, г. Архангельск, staheeva.a@narfu.ru

Вяткин Дмитрий Андреевич, старший преподаватель, Северный (Арктический) федеральный университет им. М. В. Ломоносова, г. Архангельск, d.vyatkin@narfu.ru

Stakheeva Alyona Alekseevna, Master's student, M. V. Lomonosov Northern (Arctic) Federal University, Arkhangelsk, staheeva.a@narfu.ru

Vyatkin Dmitry Andreevich, Senior Lecturer, M. V. Lomonosov Northern (Arctic) Federal University, Arkhangelsk, d.vyatkin@narfu.ru

Аннотация. Одна из важнейших задач, без которой не может существовать беспилотный автомобиль, – распознавание дорожной разметки. В данной статье рассматривается возможность распознавания линий дорожной разметки с помощью компьютерного зрения. В процессе работы над решением данной задачи был написан код для распознавания дорожной разметки на языке Python. В результате выполнения получившейся программы на видеоизображении для наглядности рисуются предполагаемые линии разметки и средняя между ними, а в

терминал программы выводится решение о повороте автомобиля в одну из сторон, основанное на значении коэффициента уравнения прямой (наклона средней линии между разметкой) и положения линий дорожной разметки прямо перед автомобилем.

Annotation. One of the most important tasks, without which an unmanned vehicle cannot exist, is the recognition of road markings. This article discusses the possibility of recognizing road marking lines using computer vision. In the process of working on solving this problem, code was written for recognizing road markings in Python. As a result of the execution of the resulting program, the intended marking lines and the average between them are drawn on the video image for clarity, and a decision on turning the car in one of the sides is output to the program terminal, based on the value of the coefficient of the equation of the straight line (the slope of the middle line between the markings) and the position of the road marking lines directly in front of the car.

Ключевые слова: автопилотирование автомобиля, дорожная разметка, компьютерное зрение, Python, OpenCV, распознавание на видеоизображении.

Keywords: car autopiloting, road markings, computer vision, Python, OpenCV, video image recognition.

Постановка задачи и входные данные

Существование и правильная работоспособность беспилотного автомобиля невозможна, если он не будет уметь распознавать линии дорожной разметки. Видеокамеры являются основной технологией для распознавания полос движения, дорожных знаков, светофоров, других транспортных средств и пешеходов. Получаемое изображение в режиме реального времени отправляется на центральный компьютер и с помощью компьютерного зрения система идентифицирует различные объекты и, основываясь на этом и на прописанных алгоритмах, принимает решения для дальнейших действий. Камеры позволяют хорошо определять окружающую среду за счет большого количества существующих библиотек обработки видеопотоков, однако это будет сложно использовать при плохих погодных условиях (сумерки, дождь и снег).

Для решения задачи распознавания дорожной разметки было решено использовать различные методы компьютерного зрения. Для написания программы будет использован язык программирования Python, так как у него существует большое количество библиотек, заточенных для работы с изображениями. Компьютерное зрение (Computer Vision, CV) – это область искусственного интеллекта, связанная с анализом изображений и видео. Она включает в себя набор методов, которые наделяют компьютер способностью «видеть» и извлекать информацию из увиденного. Системы состоят из фото- или видеокамеры и специализированного программного обеспечения, которое идентифицирует и классифицирует объекты.

В качестве входных данных будет выступать видео с камеры, установленной в машине. Пример кадра из видео, используемого с данной работе, представлен на рисунке 1.

Рисунок 1 – Кадр из используемого видео

Для решения задачи распознавания дорожной разметки на видео нужно выполнить следующее:

– кадрировать видео до рабочей области;

– получить разными способами маски с детектированной дорожной разметкой;

– определить лучшую маску или маски для различных условий;

– задать рабочую область маски для определения дорожной разметки;

– определить на левой и правой сторонах маски для определения дорожной разметки точки с наиболее белыми участками, а также вычислить средние между ними;

– получить с помощью полученных точек три уравнения прямых и построить две линии, которые должны совпадать с линиями дорожной разметки, и одну среднюю между ними;

– сделать вывод о необходимости поворота машины на основе коэффициентов уравнения средней линии;

– задать рабочую область маски для нахождения положения транспортного средства между линиями дорожной разметки;

– определить на левой и правой сторонах маски для нахождения положения транспортного средства между линиями дорожной разметки точки с наиболее белыми участками;

– сделать вывод о необходимости движения транспортного средства влево или вправо для поддержания среднего положения между линиями дорожной разметки.

Программная реализация распознавания дорожной разметки

Для начала необходимо создать проект, подключить используемые библиотеки и добавить видео для обработки, а также вывести размеры данного видео (листинг 1).

Листинг 1 – Подключение библиотек и загрузка рабочего видео

```
# Импортируем библиотеки:
import cv2 as cv
import numpy as np

# Загружаем видео для работы:
video = cv.VideoCapture('road2.mp4')

# Получение данных о размере видео:
width_orig  = video.get(3)
height_orig = video.get(4)
print('height:', height_orig, 'width:', width_orig)
```

Далее пропишем в коде о том, что делать, если видео не будет найдено или не сможет открыться. Сама рабочая программа при успешном открытии видео будет прописана в цикле While (листинг 2).

Листинг 2 – Открытие видео

```
# Уведомление об ошибке, если видео не может открыться:
if not video.isOpened():
    print('Error')
    exit()

while True:
    ret, frame = video.read() # читаем видео
    # cv.imshow("frame", frame)

    if cv.waitKey(1) & 0xFF == ord('q'):
        break
video.release()
cv.destroyAllWindows()
```

Весь дальнейший код, который будет представлен, располагается в данном цикле While. Для более точного распознавания дорожной разметки лучше кадрировать видео и оставить только тот участок, на котором она может встречаться (листинг 3). Пример уже кадрированного видео показан на рисунке 2.

Листинг 3 – Кадрирование видео

```
    v_frame = frame[330:640, 300:1280] # кадрирование видео
    mask_center = v_frame.copy() # копия обрезанного видео
    # cv.imshow("frame", v_frame)
```

Для детектирования дорожной разметки получим маски несколькими разными способами (листинг 4). Первая маска получена с помощью перевода изображения в оттенки серого и установки порогового значения. Для второй и третьей маски видеоизображение сначала надо перевести в цветовое пространство HSV. После этого второе видео переводиться в одноканальное и устанавливается пороговое значение для реагирования и преобразования изображения в черно-белую маску. Для третьего же видео устанавливаются верхний и нижний пределы

и пиксели, которые попадают в данный диапазон, в итоге на маске имеют белый цвет, а те, которые не попадают, – черный. На четвертой маске отображены края (границы) всех объектов видеоизображения. Все четыре полученные маски показаны на рисунке 3.

Рисунок 2 – Кадрированное видео

Листинг 4 – Маски для детектирования дорожной разметки

```
    # Маска, полученная с помощью перевода изображения в GRAY:
    v_grey = cv.cvtColor(v_frame, cv.COLOR_BGR2GRAY)
    _, v_threshold = cv.threshold(v_grey, 127, 255, 0)
    mask_1 = v_threshold
    # cv.imshow('mask_1', mask_1)

    # Маска, полученная с помощью перевода изображения в HSV:
    hsv = cv.cvtColor(v_frame, cv.COLOR_BGR2HSV)
    ch_b = hsv[:,:,2]
    mask_2 = np.zeros_like(ch_b)
    mask_2[ch_b > 140] = 255
    # cv.imshow('mask_2', mask_2)

    # Маска, полученная с помощью перевода изображения в HSV и установки
нижнего и верхнего пределов:
    low3 = np.array([0, 0, 150])
    high3 = np.array([225, 225, 220])
    mask_3 = cv.inRange(hsv, low3, high3)
    # cv.imshow('mask_3', mask_3)
```

```
# Маска, полученная с помощью обнаружения краев:
v_bilat = cv.bilateralFilter(v_grey, 9, 75, 75)
# cv.imshow('v_bilat', v_bilat)
v_canny = cv.Canny(v_bilat, 50, 150)
# cv.imshow('canny_bilat', v_canny)
```

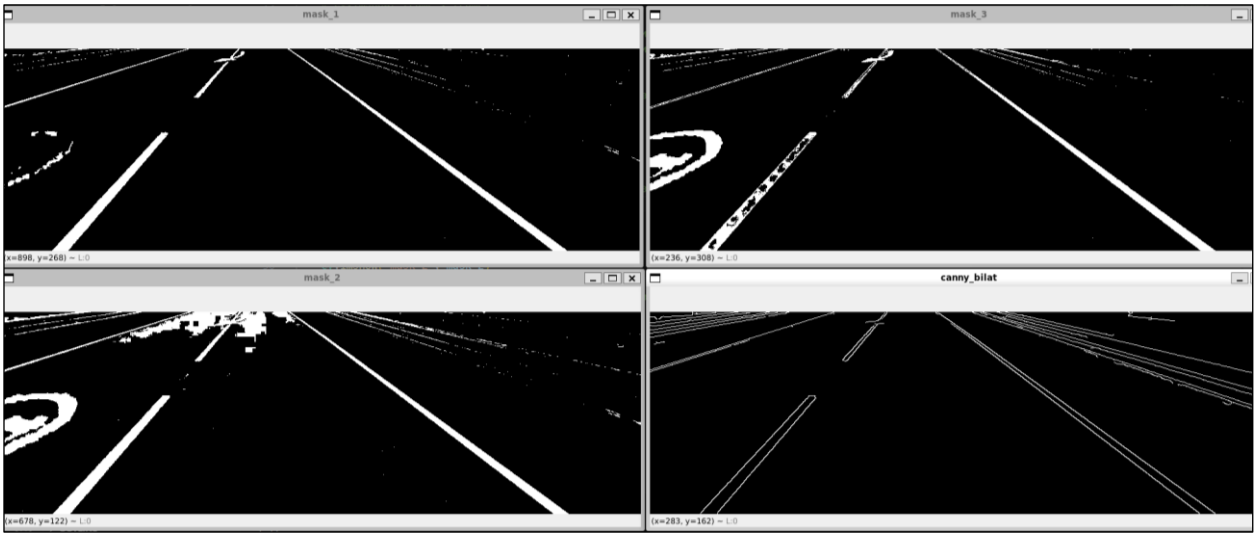

Рисунок 3 – Маски для детектирования дорожной разметки

Из всех полученных масок для лучшего результата воспользуемся либо объединенной маской из первых трех, либо, если на маске слишком много белого (т.е. много лишнего), то возьмем как итоговый результат четвертую маску с обнаруженными границами (листинг 5).

Листинг 5 – Объединенная маска для детектирования дорожной разметки

```
# Объединение масок:
allmask = np.zeros_like(v_grey)
allmask[(mask_1==255)|(mask_2==255)|(mask_3==255)]=255
dop_mask = allmask.copy()
if np.sum(allmask[:,:]) > 7000000:
    allmask = v_canny
# cv.imshow('ALL mask', allmask)
```

Для более точного определения линий дорожной разметки отсеем все лишнее по бокам видео, для этого укажем границы трапеции и оставим изображение только в ней (листинг 6). Рабочая область для маски показана на рисунке 4.

Листинг 6 – Задаем рабочую область для маски

```
# Задаем рабочую область для маски:
```

```
    tr_mask = np.zeros_like(allmask)
    trapeze = np.array([(0,310), (300, 0), (550, 0), (980,310)],
dtype=np.int32) # указываем нужные границы трапеции
    cv.fillPoly(tr_mask, [trapeze], 255)
    mask_tr_binary = cv.bitwise_and(allmask, tr_mask)
    # cv.imshow('mask_tr_binary', mask_tr_binary)
    mask_tr_orig = cv.bitwise_and(v_frame, v_frame, mask=tr_mask)
    # cv.imshow('mask_tr_orig', mask_tr_orig)
```

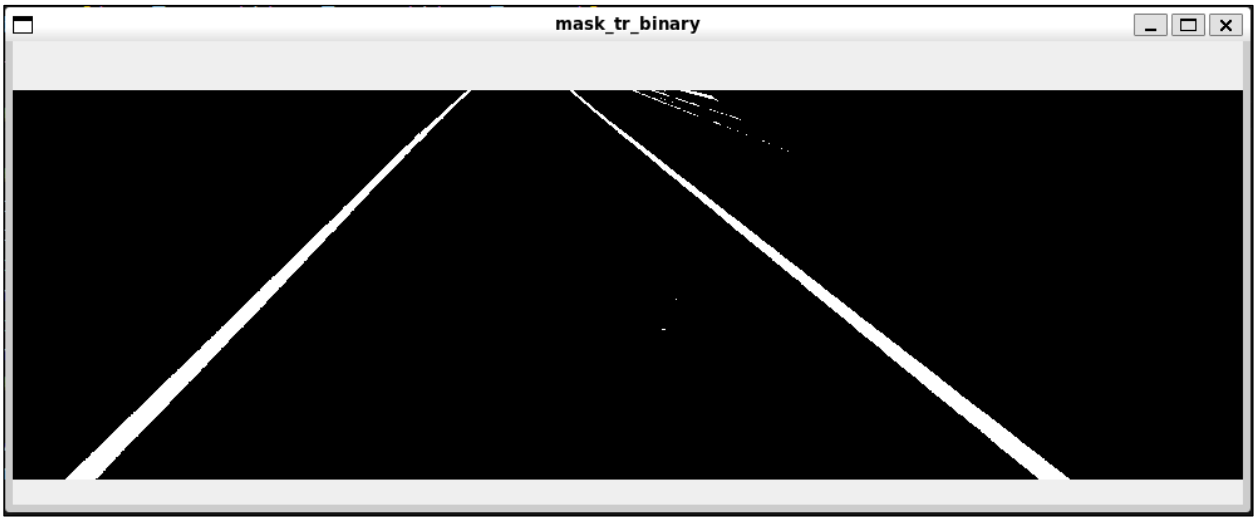

Рисунок 4 – Рабочая область для маски

Для того чтобы найти уравнения прямой для дорожной разметки и линии между ними нужно определить точки, по которым это уравнение будет строиться (листинг 7). Искать эти точки мы будем отдельно для левой и правой стороны видео, то есть для каждой полосы отдельно. Определим количество интервалов по вертикали, на которых будем искать самые светлые места маски, то есть местонахождение дорожной разметки. Чем меньше количество этих интервалов, тем менее точным будет найденное уравнение прямой, однако и слишком большое количество интервалов не даст лучший результат, а лишь будет лишь усложнять работу поиска дорожных линий. На каждом интервале отдельно в правой и левой частях видео будем искать столбец с самым большим количеством белых пикселей, а также найдем среднее арифметическое между найденными точками и все это запишем в списки.

Листинг 7 – Нахождение точек дорожной разметки для уравнений прямых

```
    # Текущие размеры видео:
```

```
    width  = 980
    height = 310
    center_axis = width//2 # середина ширины видео

    quantity = 20 # количество интервалов, на которых будут определяться
точки с самыми светлыми местами маски
    he_in = int(height/quantity)
    cent_y = []
    cent_x = []
    left_x = []
    right_x = []

    # Заполнение списка средними точками интервалов по вертикали:
    for prom in range(0, height, he_in):
        cent_y.append(int(he_in/2 + prom))

    # Поиск точек с самыми светлыми местами маски среди столбцов правой и
левой частей видео, вычисление средних точек и добавление их в списки:
    for i in range(0, len(cent_y)):
        if i == 0:
            left_i = np.argmax(np.sum(mask_tr_binary[0:cent_y[i],:],
axis=0)[:center_axis])
            right_i = np.argmax(np.sum(mask_tr_binary[0:cent_y[i],:],
axis=0)[center_axis:])+center_axis
        else:
            left_i = np.argmax(np.sum(mask_tr_binary[cent_y[i-
1]:cent_y[i],:], axis=0)[:center_axis])
            right_i = np.argmax(np.sum(mask_tr_binary[cent_y[i-
1]:cent_y[i],:], axis=0)[center_axis:])+center_axis
        left_x.append(left_i)
        right_x.append(right_i)
        cent_x.append((left_i + right_i) // 2)
```

Найдем коэффициенты уравнений трех прямых и отобразим их на видео (листинг 8). Полученный результат показан на рисунке 5.

Листинг 8 – Построение прямых, совпадающих с линиями разметки и средней между ними

```
    # Находим коэффициенты уравнений прямых:
    line_c = np.polyfit(cent_y, cent_x, 1)
    left_c = np.polyfit(cent_y, left_x, 1)
    right_c = np.polyfit(cent_y, right_x, 1)
    # parab_c = np.polyfit(cent_y, cent_x, 2)

    # Строим полученные прямые на видео:
    for ver_id in range(height):
        gor_id_o = (line_c[0]*ver_id + line_c[1])
        cv.circle(mask_center, (int(gor_id_o), int(ver_id)), 3, (0,0,0), 3)
        gor_id_l = (left_c[0]*ver_id + left_c[1])
        cv.circle(mask_center, (int(gor_id_l), int(ver_id)), 2, (0,0,255),
3)
        gor_id_r = (right_c[0]*ver_id + right_c[1])
        cv.circle(mask_center, (int(gor_id_r), int(ver_id)), 2, (0,0,255),
3)
        # gor_id_p = (parab_c[0]*ver_id**2 + parab_c[1]*ver_id +
parab_c[2])
        # cv.circle(mask_center, (int(gor_id_p), int(ver_id)), 2,
(255,0,0), 3)
    cv.imshow('mask_line', mask_center)
```

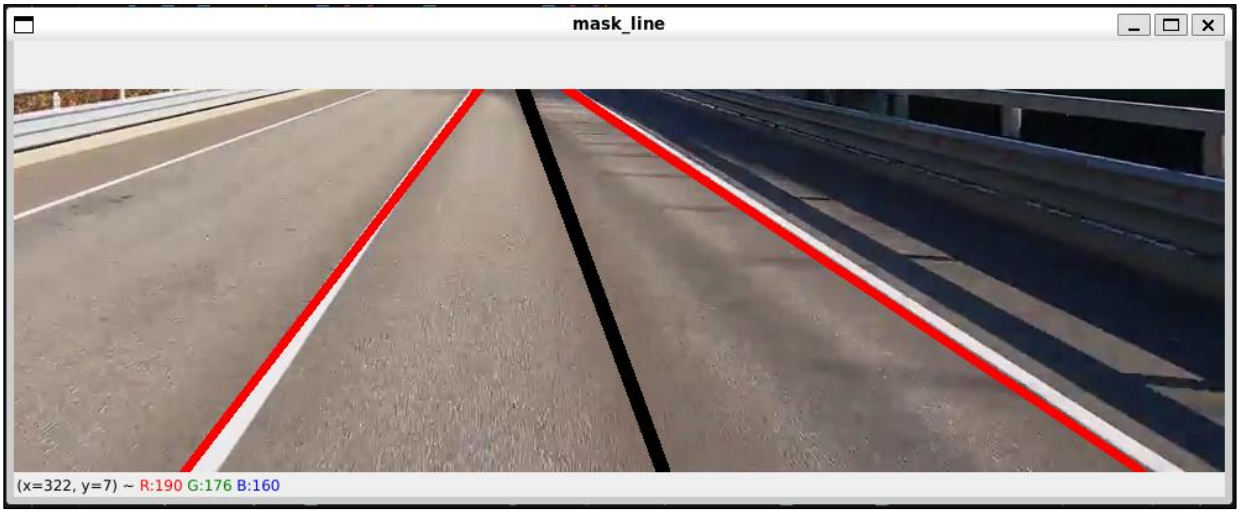

Рисунок 5 – Отображение полученных линий

Важно определять положения транспортного средства между дорожными полосами. Для этого оставим только нижнюю часть маски и по ней будем определять, где находятся полосы прямо перед машиной (листинг 9). Данная маска представлена на рисунке 6.

Листинг 9 – Определение нахождения полос прямо перед машиной

```
    # Оставляем от маски только нижнюю ее часть:
    rectr_mask = np.zeros_like(dop_mask)
    rectangle = np.array([(0,310), (0, 250), (980, 250), (980,310)],
dtype=np.int32)
    cv.fillPoly(rectr_mask, [rectangle], 255)
    mask_rectr_bin = cv.bitwise_and(dop_mask, rectr_mask)
    # cv.imshow('mask_rectr_br', mask_rectr_bin)

    # Поиск точек с самыми светлыми местами маски среди столбцов правой и
левой частей видео на нижней части маски:
    left_n = np.argmax(np.sum(mask_rectr_bin[250:310,:],
axis=0)[:center_axis])
    right_n = np.argmax(np.sum(mask_rectr_bin[250:310,:],
axis=0)[center_axis:])+center_axis

    # Определение количества белых пикселей в правой и левой частях видео
на нижней части маски:
    sum_l = np.sum(mask_rectr_bin[250:310,0:center_axis])
    sum_r = np.sum(mask_rectr_bin[250:310,center_axis:width])
```

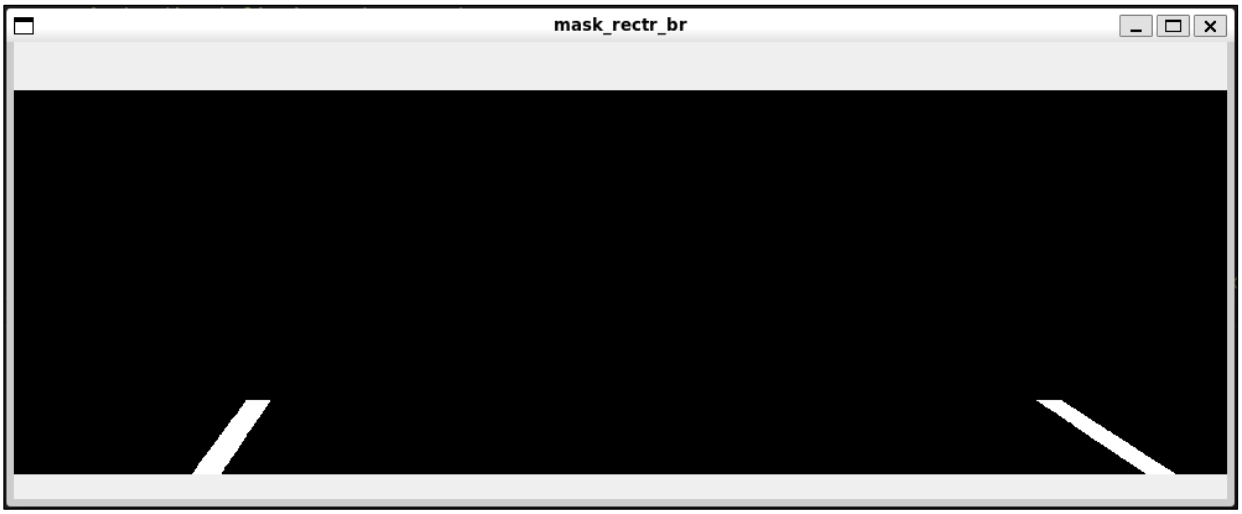

Рисунок 6 – Маска для нахождения полос прямо перед машиной

Принимать решение о повороте машины в связи с последующим направлением дороги мы будем по первому коэффициенту уравнения средней прямой, то есть по направлению и степени наклона средней линии между разметкой. Решение о движение в стороны для регулирования нахождения машины

между дорожной разметкой мы будем по положению линий рядом с ней. А общее решение о повороте машине будет приниматься таким образом, чтобы повороты не противоречили друг другу, и выводиться в терминал. Вместе с тем добавлено отображение финального видео с линиями разметки и подписями о поворотах. Код для реализации вышеперечисленных действий представлен в листинге 10.

Листинг 10 – Принятие решения о повороте машины и отображение полученного видео

```
# Создаем переменные для принятия решения о повороте:
    turn_right = 0
    turn_left = 0
    move_right = 0
    move_left = 0

    # Решение о повороте машины в зависимости от коэффициента уравнения
прямой (наклона средней линии между разметкой):
    if line_c[0] < -0.4:
        cv.putText(mask_center, "->", (890,40), cv.FONT_HERSHEY_SIMPLEX,
1.3, (0,0,255), 3)
        cv.putText(mask_center, "|", (890,60), cv.FONT_HERSHEY_SIMPLEX,
1.3, (0,255,0), 3)
        turn_right = 1
        turn_left = 0
    elif line_c[0] > 0.4:
        cv.putText(mask_center, "<-", (20,40), cv.FONT_HERSHEY_SIMPLEX,
1.3, (0,255,0), 3)
        cv.putText(mask_center, "|", (75,60), cv.FONT_HERSHEY_SIMPLEX, 1.3,
(0,255,0), 3)
        turn_left = 1
        turn_right = 0
    else:
        turn_right = 0
        turn_left = 0

    # Решение о повороте машины в зависимости от положения линий дорожной
разметки рядом с машиной:
    if sum_l < 500000 and sum_r < 500000:
```

```
        if 0 < left_n < 110 or center_axis < right_n < 850:
            cv.putText(mask_center, "<-", (20,150),
cv.FONT_HERSHEY_SIMPLEX, 1.3, (0,255,0), 3)
            move_left = 1
            move_right = 0
        elif 140 < left_n < center_axis or 880 < right_n < 980:
            cv.putText(mask_center, "->", (890,150),
cv.FONT_HERSHEY_SIMPLEX, 1.3, (0,255,0), 3)
            move_right = 1
            move_left = 0
        else:
            move_right = 0
            move_left = 0

    # Общее решение о повороте машины:
    if (move_right==1 or turn_right==1) and turn_left==0 and move_left==0:
        print('right')
        cv.putText(mask_center, "Turn right", (740,290),
cv.FONT_HERSHEY_SIMPLEX, 1.4, (255,0,0), 4)
    elif (move_left==1 or turn_left==1) and turn_right==0 and
move_right==0:
        print('left')
        cv.putText(mask_center, "Turn left", (20,290),
cv.FONT_HERSHEY_SIMPLEX, 1.4, (255,0,0), 4)
    else:
        print('stay')
        cv.putText(mask_center, "Stay put", (400,290),
cv.FONT_HERSHEY_SIMPLEX, 1.4, (255,0,0), 4)

    # Отображение финального видео с линиями разметки и подписями о
поворотах
    cv.imshow('mask_center', mask_center)
```

Тестирование программы на входном видеоизображении

Несколько кадров с результатом работы программы показаны на рисунках 7, 8, 9 и 10. На рисунке 7 средняя линяя не имеет большого наклона и автомобиль находиться примерно по середине линий разметка, поэтому ему не нужно

поворачиваться ни в одну из сторон. На рисунке 8 происходит противоречие и по итогу принимается решение не поворачиваться, так как со временем это компенсируется и автомобиль займет среднее положение между линиями дорожной разметки. На рисунке 9 машина немного выходит за рамки линий разметки и ей нужно вернуться в пределы своей полосы. На рисунке 10 дорога имеет небольшой поворот налево, поэтому и автомобилю тоже необходимо его сделать.

После тестирования данного кода на нескольких видео можно сделать вывод, что написанная программа работает довольно успешно, однако существуют некоторые факторы, влияющие на точность распознавания дорожной разметки. Малое влияние на качество работы оказывает наличие прерывистой линии разметки вместо непрерывной. Чуть большее влияние оказывают плохие погодные условия, сильное или слабое солнечное освещение, посторонние предметы (помехи) или транспортные средства, закрывающие большую часть видимой разметки.

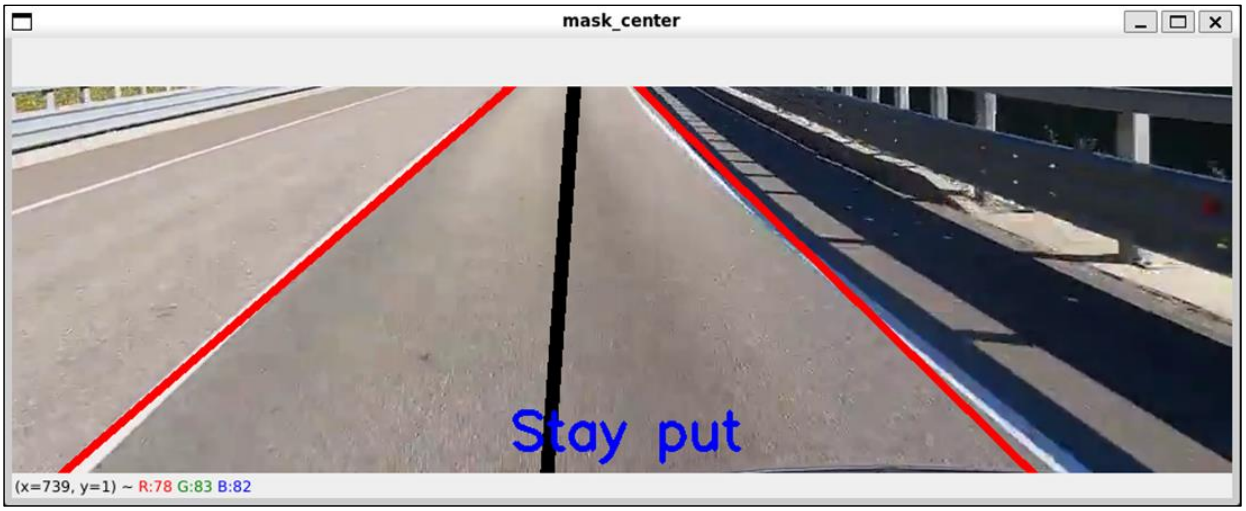

Рисунок 7 – Результат работы программы (пример 1)

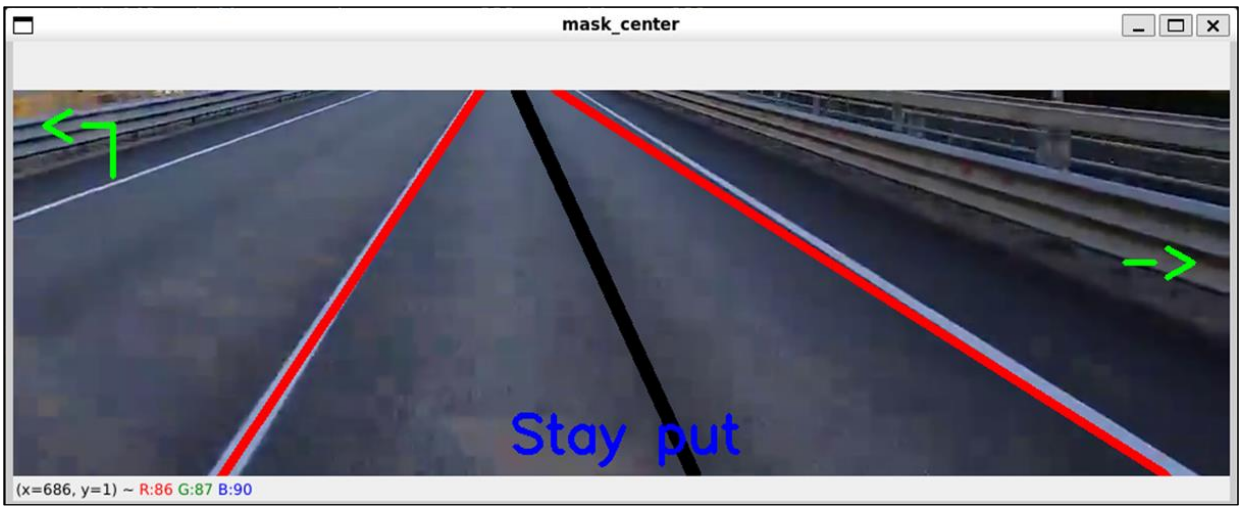

Рисунок 8 – Результат работы программы (пример 2)

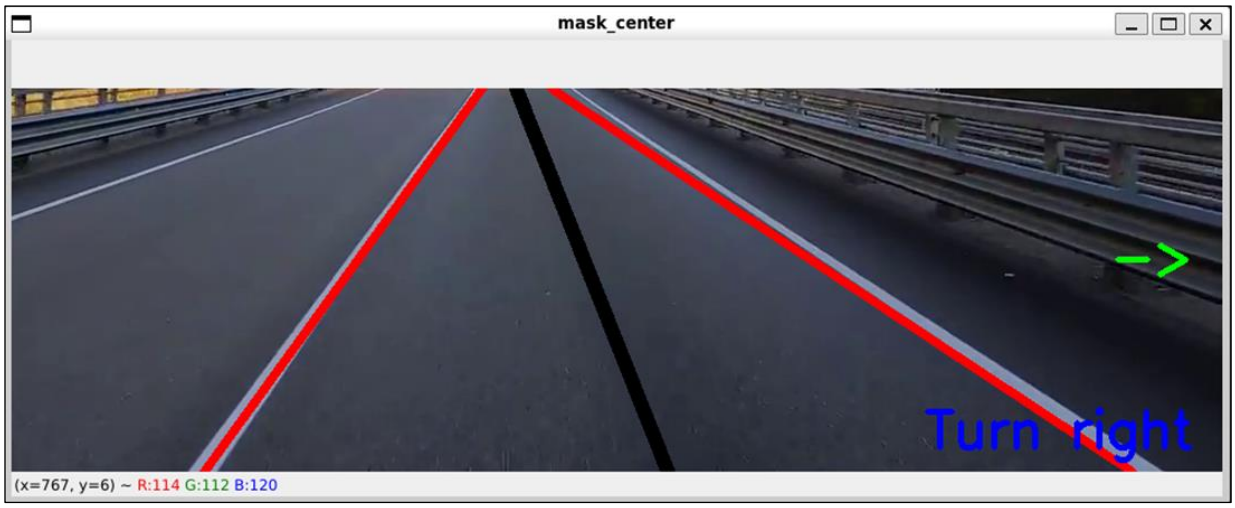

Рисунок 9 – Результат работы программы (пример 3)

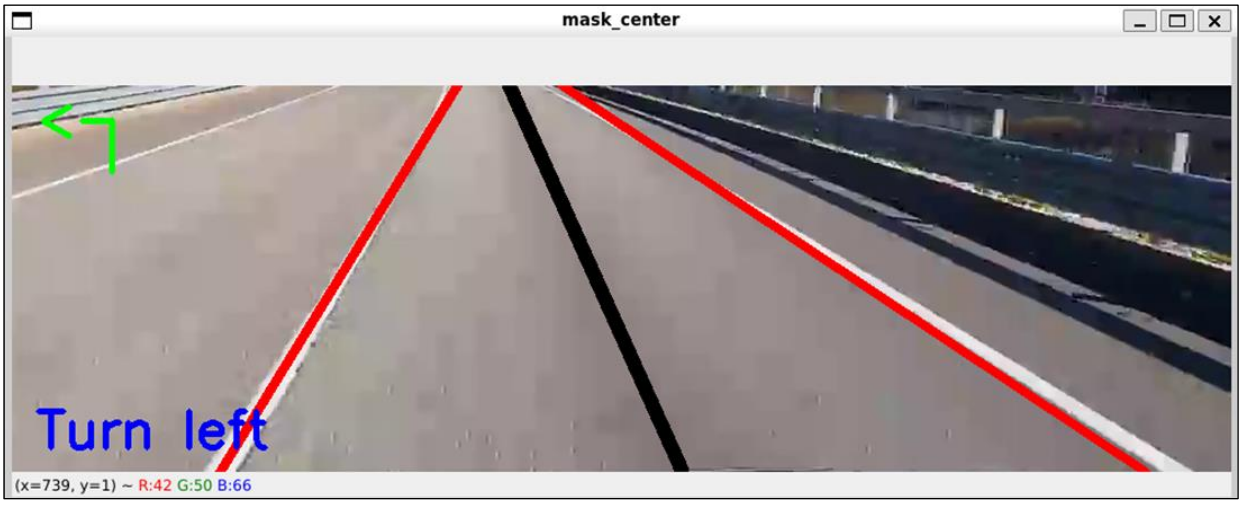

Рисунок 10 – Результат работы программы (пример 4)

Заключение

В ходе работы была написана программа для распознавания дорожной

разметки на языке Python, в результате выполнения которой на видеоизображение выводиться предполагаемые линии разметки и средняя между ними. В терминал программы выводиться решение о повороте автомобиля в одну из сторон, если он необходим в данной дорожной ситуации, основанное на значении коэффициента уравнения прямой (наклона средней линии между разметкой) и положения линий дорожной разметки рядом с машиной.

Написанная программа работает удовлетворительно и справляется в поставленными задачами, значит можно сделать вывод о том, что с помощью компьютерного зрения решение задачи распознавания дорожной разметки является возможным. Однако на качество распознавания дорожной разметки могут влиять некоторые факторы. Разрешение, стабилизация и захват камеры, которая снимает дорогу, могут в достаточной степени влиять на работоспособность. Программа не сможет корректно функционировать, если камера будет установлена так, что не все линии дорожной разметки будут помещаться на изображении. Разрешение и стабилизация будут влиять на качество входного видео и ухудшать точность распознавания. Малое влияние на качество работы оказывает наличие прерывистой линии разметки вместо непрерывной. Чуть большее влияние оказывают плохие погодные условия, сильное или слабое солнечное освещение, посторонние предметы (помехи) или транспортные средства, закрывающие большую часть видимой разметки. Решить проблему с условиями погоды можно с помощью автоматической корректировки реагирования на различные параметры изображения, например такие как яркость, насыщенность и контрастность.

Литература

1. Климов А.А., Покусаев О.Н., Куприяновский В.П., Намиот Д.Е. Архитектура автономных (беспилотных) автомобилей и инфраструктура для их эксплуатации [Электронный ресурс] / А.А. Климов, О.Н. Покусаев, В.П. Куприяновский, Д.Е. Намиот // СОВРЕМЕННЫЕ ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ И ИТ-ОБРАЗОВАНИЕ : науч. электрон. журн. – 2018. – №3(14). – С. 727–736. – Электрон. текстовые дан. – Режим доступа : https://www.elibrary.ru/item.asp?id=37031854, доступ из НЭБ «E-Library» (дата

обращения : 09.08.2023) – Загл. с экрана;

2. Назаренко М.А., Федулаева Д.Е. Компоненты системы качества беспилотных систем парковки легковых автомобилей [Электронный ресурс] / М.А. Назаренко, Д.Е. Федулаева // АКТУАЛЬНЫЕ ПРОБЛЕМЫ И ПЕРСПЕКТИВЫ РАЗВИТИЯ РАДИОТЕХНИЧЕСКИХ И ИНФОКОММУНИКАЦИОННЫХ СИСТЕМ "РАДИОИНФОКОМ-2019" : науч. электрон. журн. – 2019. – С. 437–439. – Электрон. текстовые дан. – Режим доступа : https://www.elibrary.ru/item.asp?id=42437404, доступ из НЭБ «E-Library» (дата обращения : 09.08.2023) – Загл. с экрана;
3. Суфиянов, Р.Ш. Лидар в системе обеспечения безопасности эксплуатации беспилотного автомобиля [Электронный ресурс] / Р.Ш. Суфиянов // ТЕНДЕНЦИИ РАЗВИТИЯ НАУКИ И ОБРАЗОВАНИЯ : науч. электрон. журн. – 2022. – №82-2. – С. 87–90. – Электрон. текстовые дан. – Режим доступа : https://www.elibrary.ru/item.asp?id=48056924, доступ из НЭБ «E-Library» (дата обращения : 09.08.2023) – Загл. с экрана;
4. How Do Self-Driving Cars Work? [Электронный ресурс] : [офиц. сайт] / Interesting engineering – Электрон. дан. – [2011-2023]. Режим доступа : https://interestingengineering.com/innovation/how-do-self-driving-cars-work, свободный (дата обращения : 09.08.2023). – Загл. с экрана;
5. Чебыкин И.А., Семенов С.С. Автоматизация мониторинга дорожного движения с помощью компьютерного зрения [Электронный ресурс] / И.А. Чебыкин, С.С. Семенов // ТРАНСПОРТ. ТРАНСПОРТНЫЕ СООРУЖЕНИЯ. ЭКОЛОГИЯ : науч. электрон. журн. – 2020. – С. 52-60. – Электрон. текстовые дан. – Режим доступа : https://elibrary.ru/item.asp?id=44414468, доступ из НЭБ «E-Library» (дата обращения : 09.08.2023) – Загл. с экрана;
6. Шевченко, А.С. Технология LIDAR в сфере беспилотных автомобилей [Электронный ресурс] / А.С. Шевченко // ИННОВАЦИОННЫЙ ПОТЕНЦИАЛ РАЗВИТИЯ ОБЩЕСТВА: ВЗГЛЯД МОЛОДЫХ УЧЕНЫХ : науч. электрон. журн. – 2021. – С. 162–164. – Электрон. текстовые дан. – Режим доступа : https://elibrary.ru/item.asp?id=47409388, доступ из НЭБ «E-Library»

(дата обращения : 09.08.2023) – Загл. с экрана;

7. Простой алгоритм распознавания дорожной разметки [Электронный ресурс] : [офиц. сайт] / Reg – Москва, [2023?]. – Электрон. дан. – Режим доступа : https://www.reg.ru/blog/simple-algorithm-for-road-marking-detection/, свободный (дата обращения : 09.08.2023). – Загл. с экрана;
8. Введение в OpenCV применительно к распознаванию линий дорожной разметки [Электронный ресурс] : [офиц. сайт] / Хабр. – Москва, [2023?]. – Электрон. дан. – Режим доступа : https://habr.com/ru/companies/newprolab/articles/328422/, свободный (дата обращения : 09.08.2023). – Загл. с экрана;
9. Классы судов Российского Морского Регистра [Электронный ресурс] : [офиц. сайт] / Виктория. – Москва, [2023?]. – Электрон. дан. – Режим доступа : https://victoria.lrit.ru/class/rs_class_info.htm, свободный (дата обращения : 09.08.2023). – Загл. с экрана;
10. Друки А.А., Кружков Д.С. Разработка программного обеспечения распознавания дорожной полосы на видеопоследовательностях [Электронный ресурс] / А.А. Друки, Д.С. Кружков // МОЛОДЕЖЬ И СОВРЕМЕННЫЕ ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ: науч. электрон. журн. – 2021. – С. 25–26. – Электрон. текстовые дан. – Режим доступа : https://elibrary.ru/item.asp?id=46394693, доступ из НЭБ «E-Library» (дата обращения : 09.08.2023) – Загл. с экрана.

References

1. Klimov A.A., Pokusaev O.N., Kupriyanovsky V.P., Namiot D.E. Architecture of autonomous (unmanned) cars and infrastructure for their operation [Electronic resource] / A.A. Klimov, O.N. Pokusaev, V.P. Kupriyanovsky, D.E. Namiot // MODERN INFORMATION TECHNOLOGIES AND IT EDUCATION : scientific. electron. journal. – 2018. – №3(14). – Pp. 727-736. – Electron. text data. – Access mode : https://www.elibrary.ru/item.asp?id=37031854 , access from the NEB "E-Library" (accessed : 09.08.2023) – Title from the screen;
2. Nazarenko M.A., Fedulaeva D.E. Components of the quality system of unmanned

passenger car parking systems [Electronic resource] / M.A. Nazarenko, D.E. Fedulaeva // ACTUAL PROBLEMS AND PROSPECTS OF DEVELOPMENT OF RADIO ENGINEERING AND INFOCOMMUNICATION SYSTEMS "RADIOINFOCOM-2019" : scientific electron. journal. – 2019. – pp. 437-439. – Electron. text data. – Access mode : https://www.elibrary.ru/item.asp?id=42437404 , access from the NEB "E-Library" (accessed : 09.08.2023) – Title from the screen;

3. Sufiyanov, R.S. Lidar in the system of ensuring the safety of operation of an unmanned vehicle [Electronic resource] / R.S. Sufiyanov // TRENDS IN THE DEVELOPMENT OF SCIENCE AND EDUCATION : scientific electron. journal. – 2022. – No.82-2. – pp. 87-90. – Electron. text data. – Access mode : https://www.elibrary.ru/item.asp?id=48056924 , access from the NEB "E-Library" (accessed : 09.08.2023) – Title from the screen;
4. How Do Self-Driving Cars Work? [Electronic resource] : [ofic. website] / Interesting engineering – Electron. dan. – [2011-2023]. Access mode : https://interestingengineering.com/innovation/how-do-self-driving-cars-work , free (accessed : 09.08.2023). – Blank from the screen;
5. Chebykin I.A., Semenov S.S. Automation of traffic monitoring using computer vision [Electronic resource] / I.A. Chebykin, S.S. Semenov // TRANSPORT. TRANSPORT FACILITIES. ECOLOGY : scientific electron. journal. – 2020. – pp. 52-60. – Electron. text data. – Access mode : https://elibrary.ru/item.asp?id=44414468 , access from the NEB "E-Library" (accessed : 09.08.2023) – Title from the screen;
6. Shevchenko, A.S. LIDAR technology in the field of unmanned vehicles [Electronic resource] / A.S. Shevchenko // INNOVATIVE POTENTIAL OF SOCIETY DEVELOPMENT: THE VIEW OF YOUNG SCIENTISTS : scientific electron. journal. – 2021. – pp. 162-164. – Electron. text data. – Access mode : https://elibrary.ru/item.asp?id=47409388 , access from the NEB "E-Library" (accessed : 09.08.2023) – Title from the screen;
7. A simple algorithm for recognizing road markings [Electronic resource] : [ofic. website] / Reg – Moscow, [2023?]. – Electron. dan. – Access mode :

https://www.reg.ru/blog/simple-algorithm-for-road-marking-detection /, free (accessed : 09.08.2023). – Blank from the screen;

8. Introduction to OpenCV in relation to the recognition of road marking lines [Electronic resource] : [ofic. website] / Habr. – Moscow, [2023?]. – Electron. dan. – Access mode : https://habr.com/ru/companies/newprolab/articles/328422 /, free (accessed : 09.08.2023). – Blank from the screen;
9. Classes of vessels of the Russian Maritime Register [Electronic resource] : [ofic. website] / Victoria. – Moscow, [2023?]. – Electron. dan. – Access mode : https://victoria.lrit.ru/class/rs_class_info.htm , free (accessed : 09.08.2023). – Blank from the screen;
10. Druki A.A., Kruzhkov D.S. Development of software for recognizing a road lane on video sequences [Electronic resource] / A.A. Druki, D.S. Kruzhkov // YOUTH AND MODERN INFORMATION TECHNOLOGIES: scientific electron. journal. – 2021. – pp. 25-26. – Electron. text data. – Access mode : https://elibrary.ru/item.asp?id=46394693 , access from the NEB "E-Library" (accessed : 09.08.2023) – Title from the screen.

Для цитирования: Стахеева А.А., Вяткин Д.А. Распознавание дорожных знаков с использованием сверточной нейронной сети// Научный сетевой журнал «Столыпинский вестник» №8/2023.