

**Содержание**

[Введение 3](#_Toc108913602)

[Доработка системы 4](#_Toc108913603)

[Код 9](#_Toc108913604)

[Вывод 13](#_Toc108913605)

[Список литературы 14](#_Toc108913606)

# **Введение**

Учебная технологическая практика - систематическая деятельность студентов вуза, основанная на сборе, обработке и обобщении научного материала посредством использования накопленной теоретической и практической базы, индивидуальной характеристики мышления и работы.

Согласно ФГОС (Федеральный государственный образовательный стандарт) учебная технологическая практика студентов магистратуры является обязательным разделом образовательной программы магистратуры и направлена на формирование общекультурных и профессиональных компетенций выпускников, способствующих их успешному профессиональному росту.

Учебная технологическая практика проходит с 22 июня по 19 июля 2022 года в Нижегородском государственном архитектурно-строительном университете на кафедре Информационных систем и технологий.

Целью учебной технологической практики является автоматизация нахождения на видеоматериале с видеокамеры кодового сигнала светодиода.

Во время научно-исследовательской практики требуется:

* изучить библиотеку OpenCV
* разработать систему, которая будет находить на видеоматериале с видеокамеры светодиод, пульсирующий в определённой кодовой последовательности.

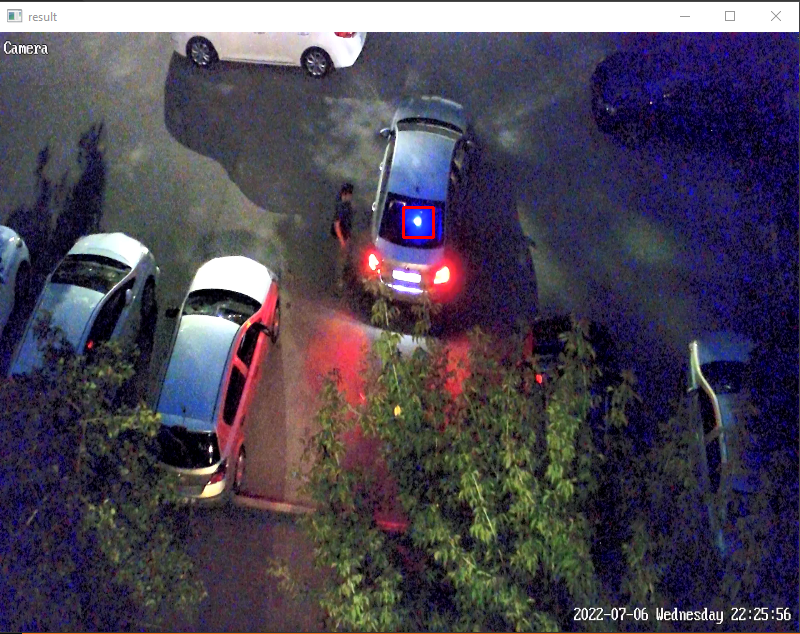
# **Доработка системы**

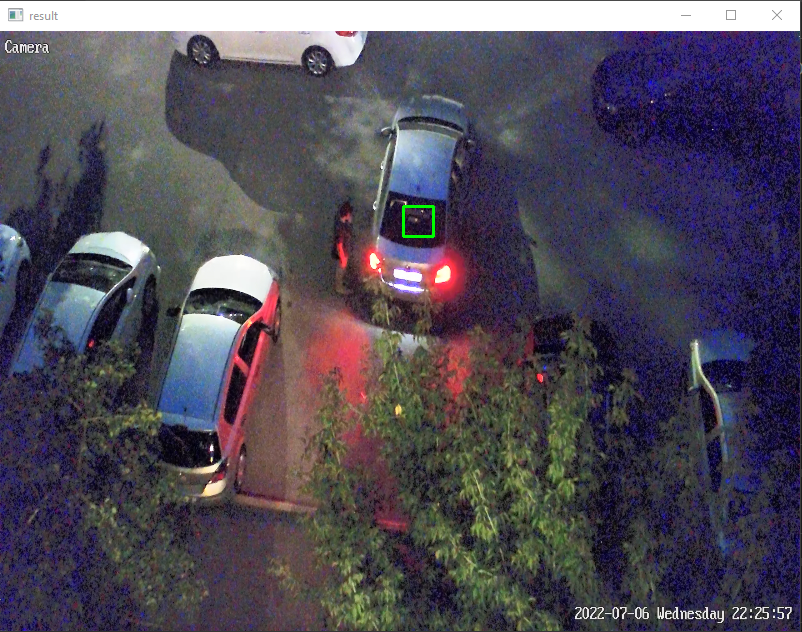
В течении прошлой научно-исследовательской практики была разработана системы по поиску светодиода на видеоматериале с камеры видеонаблюдения, используя компьютерное зрение и библиотеку OpenCV.

На данной учебной технологической практики система будет доработана, путём добавления алгоритма по считыванию последовательности пульсации синего светодиода (мощность 3 Вт), установленного в салоне машины.

Алгоритм включает в себя заполнение массива нулями и единицами в соответствии последовательности, которая запрограммирована в пульсирующий светодиод. Пульсация светодиода содержит в себе кодовую последовательность из длинных и коротких свечений. Кодовая последовательность состоит из одного длинного свечения и трёх коротких. В кодовой последовательности информацию несут паузы: длинные и короткие. Длительность сигнала одинаковая, а для отделения одной серии посылок от другой каждая серия начинается с сигнала утроенной длительности. Также алгоритм содержит распознавание данной кодовой последовательности на видеоматериале с видеокамеры.

Результат работы алгоритма отмечается зелёной областью в окне воспроизведения видеоматериала, которая показывает расположение синего светодиода с нужной кодовой последовательностью, проверяя каждый кадр на наличие светодиода [1] (Рисунок 1 и 2).

Рисунок 1 – Обозначение расположения светодиода синего цвета.

Рисунок 2 – Обозначение расположения светодиода синего цвета с заданной кодовой последовательностью.

Данный алгоритм показывает максимальную эффективность только в ночное время. Днём такой же светодиод слабо виден (Рисунок 3), а когда машина со светодиодом стоит вдали, то сигнал светодиода практически не виден (Рисунок 4).

Рисунок 3 – Свечение светодиода днём, на близком расстоянии

Рисунок 5 – Свечение светодиода днём, на дальнем расстоянии

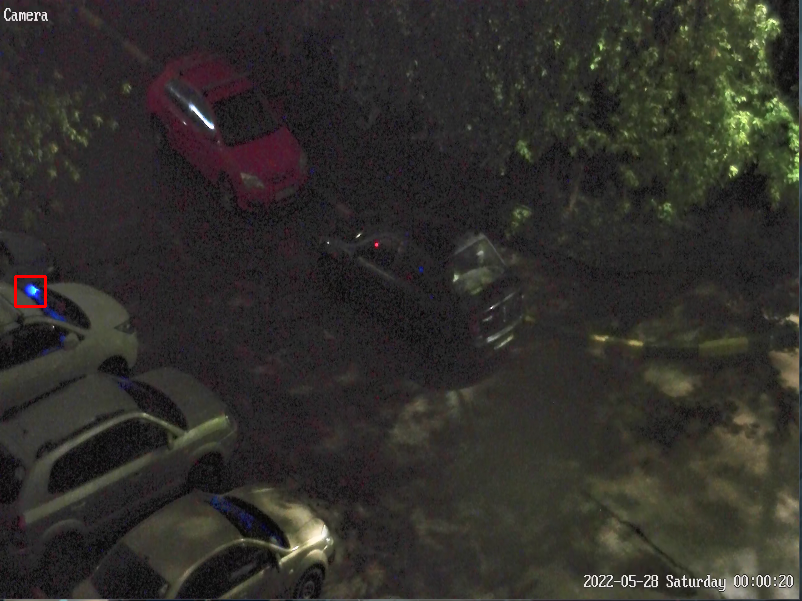
Проведём эксперимент для проверки алгоритма на правильное распознавание кодовой последовательности, запустив алгоритму видеоматериал с светодиодами от сигнализации нескольких машин (Рисунок 6 и 7).

Рисунок 6 – Обнаружение светодиода сигнализации

Рисунок 7 – Обнаружение светодиода сигнализации

Проведя эксперимент, можно заметить, что система исправно обнаруживает синие светодиоды и помечает их красным квадратом, но так как они не соответствуют кодовой последовательности, пометка зелёным квадратом не происходит. Исходя из это можно сделать вывод, что алгоритм не реагирует на ложные последовательности.

Однако алгоритм не всегда может определить нужную кодовую последовательность. Это связано с тем, что поступающий видеоматериал может быть не корректно передан с видеокамеры, где могут быть потеряны кадры, по которым проверяется кодовая последовательность.

Есть несколько решений этой проблемы:

* Сокращение количества кадров на проверку

Можно ограничить количество кадров, на которых будет проверяться кодовая последовательность, однако алгоритм может найти другие светодиоды, у которых кодовая последовательность частично совпадает и не принадлежит нужной нам машине.

* Использование нейронной сети

Нейронную сеть можно использовать для избегания случаев потери кадров, путём обучения на множестве примерах работы пересылки кодовой последовательности светодиодом

* Использование ИК-диода и ИК-камеры

В эксперименте вместе с синим светодиодом также использовался ИК-диод такой же мощности, однако его практически не было видно, так как обычная камера не может охватить весь ИК диапазон. Можно дополнительно приобрести ИК-камеру для обнаружения ИК-диода, однако это приведёт к дополнительным затратам для работы системы.

* Использование ИК-диода и ИК-датчика

ИК-датчик может считывать посылку ИК-диода в независимости от времени суток и точно определять нужную посылку (Работа ИК-датчик подробно расписана в отчёте осенней научно-исследовательской практики 2021 года). Так же это решения является дешёвым и предположительно исправно работающим. К тому же данное решение снимет с сервера нагрузку, которая возникая из-за постоянной обработки видеоматериала с видеокамеры.

# **Код**

import cv2

import numpy as np

from array import \*

if \_\_name\_\_ == '\_\_main\_\_':

def callback(\*arg):

print (arg)

cv2.namedWindow( "result" )

#1.3, 1.5, 0.5, 0.5

fps=10

CountFrame = 0

LongDiod = 1.3

LongDark = 1.5

ShortDiod = 0.5

ShortDark = 0.5

CountLongDiod = int(fps\*LongDiod)

CountLongDark = int(fps\*LongDark)

CountShortDiod = int(fps\*ShortDiod)

CountShortDark = int(fps\*ShortDark)

x1 = 0

y1 = 0

cap = cv2.VideoCapture('2.mp4')

# HSV фильтр для синих объектов

#0, 0, 250

#255, 85, 255

CodeSequence = array('i', [])

for i in range(CountLongDiod):

CodeSequence.append(1)

for i in range(CountLongDark):

CodeSequence.append(0)

for i in range(CountShortDiod):

CodeSequence.append(1)

for i in range(CountShortDark):

CodeSequence.append(0)

for i in range(CountShortDiod):

CodeSequence.append(1)

for i in range(CountShortDark):

CodeSequence.append(0)

for i in range(CountShortDiod):

CodeSequence.append(1)

print(CodeSequence)

print("\n")

print(len(CodeSequence))

flug = True

while True:

flag, img = cap.read()

img = cv2.resize(img , (800, 600))

#crop\_img = img[10:200, 300:500]

# преобразуем RGB картинку в HSV модель

hsv = cv2.cvtColor(img, cv2.COLOR\_BGR2RGB )

hsv\_min = np.array((0, 0, 250), np.uint8)

hsv\_max = np.array((255, 85, 255), np.uint8)

# применяем цветовой фильтр

thresh = cv2.inRange(hsv, hsv\_min, hsv\_max)

# вычисляем моменты изображения

moments = cv2.moments(thresh, 1)

dM01 = moments['m01']

dM10 = moments['m10']

dArea = moments['m00']

# будем реагировать только на те моменты,

# которые содержать больше 100 пикселей

if dArea > 5:

x = int(dM10 / dArea)

y = int(dM01 / dArea)

x1 = x

y1 = y

cv2.rectangle(img, (x-15, y-15), (x+15, y+15), (0,0,255), 2)

if flug:

print(CountFrame)

if dArea > 5:

if CodeSequence[CountFrame] == 1:

CountFrame += 1

else:

CountFrame = 0

else:

if CodeSequence[CountFrame] == 0:

CountFrame += 1

else:

CountFrame = 0

if CountFrame==len(CodeSequence):

flug = False

else:

cv2.rectangle(img, (x1-15, y1-15), (x1+15, y1+15), (0,255,0), 2)

if CountFrame != 70:

CountFrame +=1

else:

CountFrame = 0

flug = True

cv2.imshow('result', img)

cv2.imshow('obj', thresh)

ch = cv2.waitKey(50)

if ch == 27:

break

cap.release()

cv2.destroyAllWindows()

# **Вывод**

За время прохождения учебной технологической практики была более подробна изучена библиотека OpenCV.

В ходе работы система, которая находит на видеоматериале с видеокамеры светодиод сигнализации машины, была доработана алгоритмом для определения светодиода по кодовой последовательности. Однако у данной системы есть недостатки: днём камера практически не может обнаружить светодиод для его обработки алгоритмом, а также возможна потеря кадров, от чего алгоритм не может сопоставить видеоматериал с записанной кодовой последовательностью. Поэтому данная система не является пригодной к эксплуатации.

Дальнейшее исследование будет сосредоточена на реализации нахождения определённой машины с помощью ИК-диода и ИК-датчика.

# **Список литературы**

1. OpenCV [Электронный ресурс] – URL: https://blog.skillfactory.ru/glossary/opencv/. (Дата обращения: 24.06.2022)