МИНОБРНАУКИ РОССИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕУЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ АРХИТЕКТУРНО-СТРОИТЕЛЬНЫЙ УНИВЕРСИТЕТ» (ННГАСУ)

Факультет Инженерно-экологических систем и сооружений

Кафедра Информационных систем и технологий

Направление подготовки/специальность:

09.04.02 Информационные системы и технологии

Направленность (профиль)/специализация: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_ Технология разработки информационных систем\_\_\_\_\_\_\_\_\_\_\_

**ОТЧЕТ**

по прохождению производственной научно-исследовательской практики

Выполнил:

Студент М.ИС-11п \_\_\_\_\_\_\_\_\_\_\_\_\_К.Д. Емелин

(подпись)

Руководитель практики от кафедры

Канд. техн. наук,

доцент каф. ИСТ \_\_\_\_\_\_\_\_\_\_\_\_Д.И. Кислицын

(подпись)

Отчет защищен с оценкой: \_\_\_\_\_\_\_\_\_\_\_\_

Дата защиты

«\_\_\_» \_\_\_\_\_\_\_\_\_\_2022 г.

г. Н.Новгород, 2022 год

**Содержание**

[Введение 3](#_Toc104767309)

[Компьютерное зрение 4](#_Toc104767310)

[Разработка системы 8](#_Toc104767311)

[Код 14](#_Toc104767312)

[Вывод 18](#_Toc104767313)

[Список литературы 19](#_Toc104767314)

# **Введение**

Научно-исследовательская работа (НИР) - 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) научно-исследовательская работа студентов магистратуры является обязательным разделом образовательной программы магистратуры и направлена на формирование общекультурных и профессиональных компетенций выпускников, способствующих их успешному профессиональному росту.

Производственная научно-исследовательская практика проходит с 9 февраля по 31 мая 2022 года в Нижегородском государственном архитектурно-строительном университете на кафедре Информационных систем и технологий.

Целью производственной научно-исследовательской практики является автоматизация нахождения на видеоматериале с видеокамеры ,,,,,,,,светодиод .

Во время научно-исследовательской практики требуется:

* изучить принцип работы компьютерного зрения
* изучить библиотеку OpenCV
* разработать систему, которая будет находить на видеоматериале с видеокамеры светодиод сигнализации машины.

# **Компьютерное зрение**

Компьютерное зрение — это научное направление в области искусственного интеллекта и связанные с ним технологии получения изображений объектов реального мира, их обработки и использование полученных данных для решения разного рода прикладных задач без участия (полного или частичного) человека.

Все задачи компьютерного зрения сводятся к анализу изображения или видеопотока (По сути представляющего из себя набор сменяющихся изображений), на котором требуется прежде всего выделить фрагмент, содержащий необходимую информацию. Для выделения обычно используют или прямоугольную область, которая ограничивает исходный фрагмент, или просто выделяют пиксели, принадлежащие ему.

Задача идентификации состоит в том, чтобы классифицировать изображение целиком. Для этого на изображении выделяются ключевые области и по ним происходит классификация, например, с помощью решающих деревьев, или сверточных нейронных сетей.

Задача распознавания объектов состоит в том, чтобы по изображению суметь выделить на нем некоторый набор объектов. Пока задача не решена в общем случае – алгоритм не может классифицировать случайные объекты на изображении. Однако способен распознавать заранее заученный набор объектов с достаточно высокой точностью (Рисунок 1).

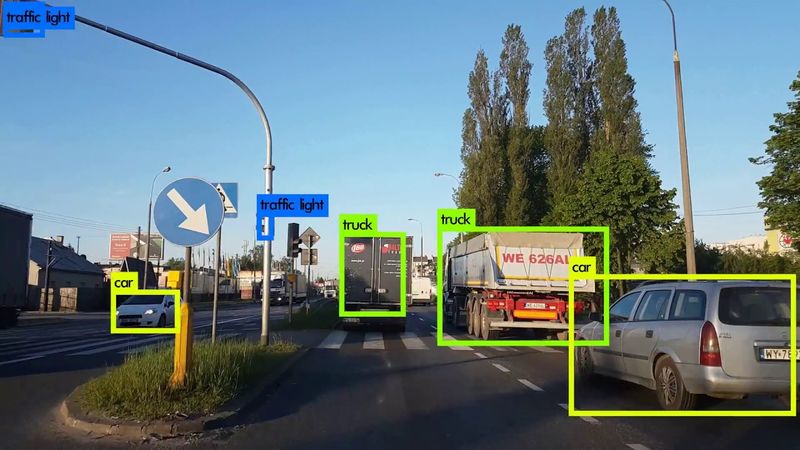


Рисунок 1 – Пример работы компьютерного зрения

Самым простым методом детекции объектов является метод скользящего окна методом R-CNN(англ. Regions with Convulational Neural Network - Выделение регионов с помощью свертоных сетей), при котором мы проходимся некоторым окном фиксированного размера по каждому кусочку картинки, и применяем к нему простой классификатор, обученный распознавать заранее определенный набор объектов. Модификации этого метода, такие как Faster R-CNN применяются до сих пор [1].

Самым популярным и известным продуктом для реализации компьютерного зрения является библиотека с открытыми исходными кодами OpenCV (Open Source Computer Vision Library), доступная для настольных и мобильных систем. Написана на C++, но существует также для Python, JavaScript, Ruby и других языков программирования. Работает на Windows, Linux и MacOS, iOS и Android.

OpenCV может использоваться везде, где нужно компьютерное зрение. Эта отрасль IT работает с технологиями, которые позволяют устройству «увидеть», распознать и описать изображение. Компьютерное зрение дает точную информацию о том, что изображено на картинке, с описанием, характеристиками и размерами (с определенной степенью достоверности).

Также библиотека работает с машинным обучением — отраслью, которая обучает алгоритмы действовать тем или иным образом.

Функции OpenCV:

1. Работа со структурами данных

Для хранения и работы с изображениями OpenCV использует векторы и скаляры, матрицы и диапазоны. Они позволяют проводить математические преобразования, ориентироваться по изображению и выполнять множество других действий.

1. Видоизменение изображений

С помощью OpenCV с картинкой можно работать как в графическом редакторе: обрезать, увеличивать или уменьшать, вращать. В основном программисты используют эту возможность для предварительной подготовки картинки перед ее расшифровкой — например, обрезают ненужные части.

1. Добавление эффектов

Картинку можно сделать в оттенках серого или полностью черно-белой. Это важно для алгоритмов распознавания, которые работают с обесцвеченными изображениями. Можно изменять цветовой тон, размывать, сглаживать или геометрически изменять картинку.

1. Рисование поверх изображения

На картинку можно нанести линии и геометрические фигуры, сделать подпись, например, чтобы выделить найденное программой лицо. Часто это используется в мобильных приложениях для камеры: квадрат вокруг лица человека во время съемки означает, что программа распознала его.

1. Распознавание объектов

Для распознавания элементов в OpenCV используются очертания объектов, сегментация по цветам, встроенные методы распознавания, которые можно настраивать в зависимости от объекта и чувствительности алгоритма.

1. Работа с видеороликами

Новые версии библиотеки поддерживают работу не только с картинками, но и с видео. Они могут считывать ролики с использованием кодеков, анализировать происходящее в них, отслеживать движения и элементы. Это полезно, например, при программировании движущегося робота или создании ПО для камеры видеонаблюдения [2].

# **Разработка системы**

Система состоит из двух Python скриптов Find\_Color.py и Find\_Diode.py.

Видеоматериал для работы системы берётся с камеры видеонаблюдения.

Скрипт Find\_Color.py отвечает за настройку цветового фильтра, что поможет определить цвет светодиода, который нам нужно отследить. Настройка происходит с помощью функции OpenCV для выделения цветового диапазона.

Функция inRange позволяет наложить на кадр цветовой фильтр в заданном диапазоне.

inRange( кадр, цвет 1, цвет 2 )

кадр — изображение, на которое мы накладываем фильтр;

цвет 1 — начальный цвет диапазона;

цвет 2 — конечный цвет диапазона.

Чтобы выделить мерцание диода нам нужен именно диапазон цветов, а не один конкретный. Так как нам нужно распознавать диод и ночью, и днём, и в любые погодные условия. Значит нам интересен и светло-синий и темно-синий цвет, а также все оттенки между ними.

Для создания подходящего фильтра библиотека OpenCV позволяет в режиме реального времени настраивать все три компонента HSV (оттенок, насыщенность, значение) картинки.

Пример работы скрипта:

С камеры видео наблюдения берётся видео поток и отображается в оконном приложении “Origin” (Рисунок 2).

Рисунок 2 – Пример видеоматериала

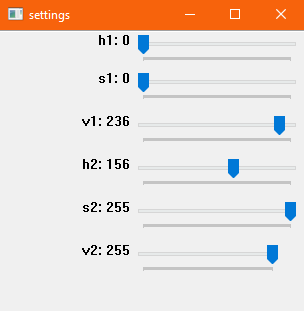
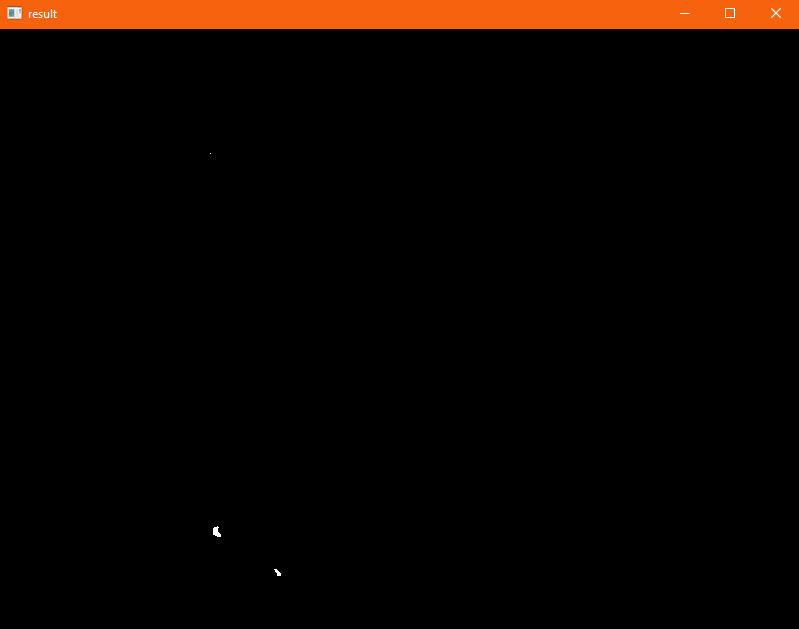
На видео мигают синие светодиоды, их нам и нужно выделить. В открывшемся окне “settings” (Рисунок 3) двигаем ползунки так чтобы в окне “result” остались только мигающие светодиоды (Рисунок 4).

Рисунок 3 – Окно настройки HSV

Рисунок 4 – Видеоматериал после наложения фильтра

Далее вписываем значения из окна настройки HSV в строки кода скрипта Find\_Diode.py:

hsv\_min = np.array((0, 0, 236), np.uint8)

hsv\_max = np.array((158, 255, 255), np.uint8)

Скрипт Find\_Diode.py отвечает за поиск и обозначение светодиодов. Здесь используется алгоритм вычисления моментов. Момент изображения — это суммарная характеристика пятна, представляющая собой сумму всех точек (пикселей) этого пятна. При этом, имеется множество подвидов моментов, характеризующие разные свойства изображения.

Например, момент нулевого порядка m00 — это количество всех точек, составляющих пятно. Момент первого порядка m10 представляет собой сумму X координат точек, а m01 — сумму Y координат. Имеются также моменты m11, m20, m02, m22 и т.д.

Формула для вычисления моментов очень проста, и мы можем посчитать пиксели вручную. Однако, стандартные функции OpenCV написаны на языках более низкого уровня, чем python, и работают быстрее. Воспользуемся стандартной функцией для вычисления моментов кадра:

moments( кадр, двоичный )

Аргумент кадр представляет собой нашу пред обработанную картинку. Аргумент двоичный определяет то, как алгоритм будет вычислять вес каждой точки. Предыдущий метод inRange дал нам черно-белую картинку, в которой пиксели могут быть черными, белыми, а могут быть и серыми. Так вот, если аргумент двоичный равен 1, то вес всех точек с цветом, отличным от нуля будет равен единице. В противном случае, вес черной точки будет равен 0, а белой точки — 255.

Функция moments вернет нам массив моментов вплоть до третьего порядка. Но для вычисления координат центра пятна нам потребуются только моменты первого порядка m01 и m10, а также момент m00.

dM01 = moments['m01']

dM10 = moments['m10']

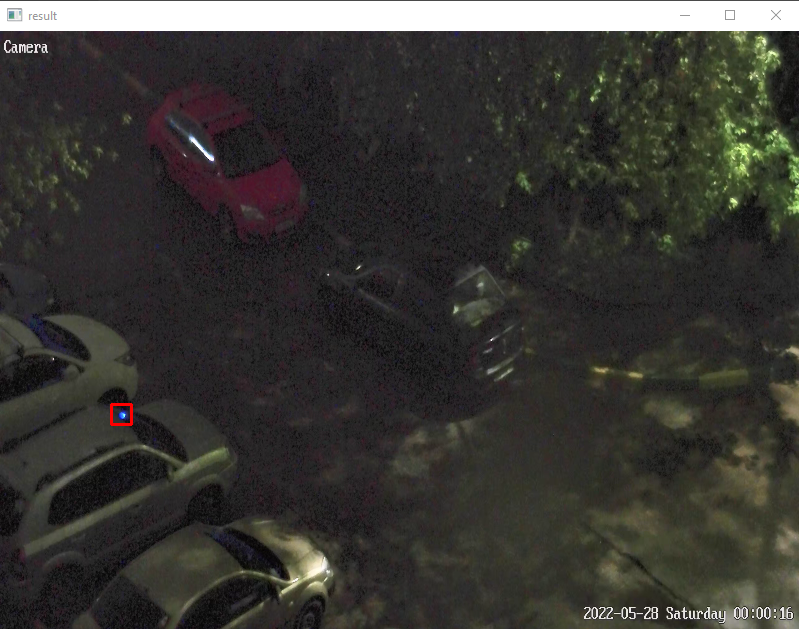
dArea = moments['m00']

Чтобы получить координаты X и Y искомого пятна, нам следует поделить полученные моменты m10 и m01 на нулевой момент m00. Таким образом мы найдем средние координаты X и Y всех точек, а это и есть центр пятна.

x = int(dM10 / dArea)

y = int(dM01 / dArea)

Также на основе координат отобразим для наглядности в открывшемся окне с видеопотоком квадраты, обозначающие светодиоды (Рисунок 5 и 6).

Рисунок 5 – Детектирование светодиода в ночное время

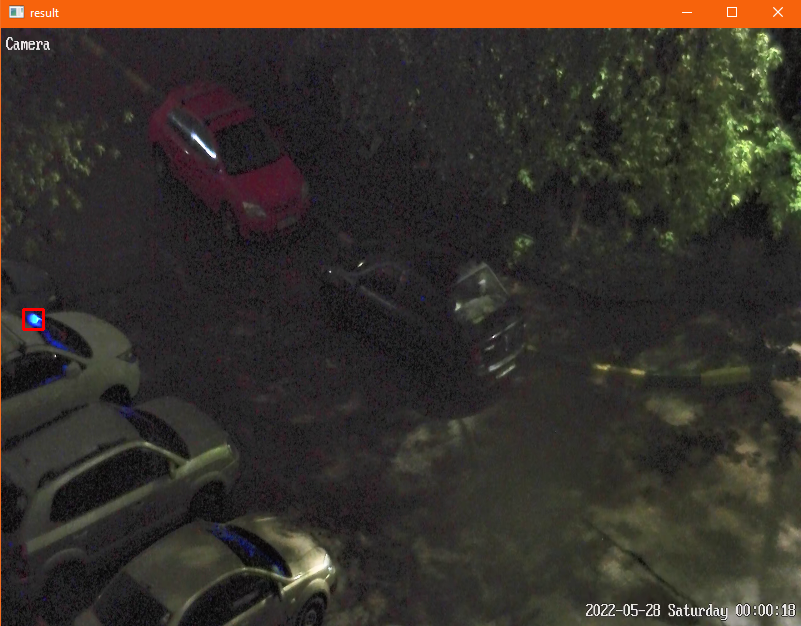
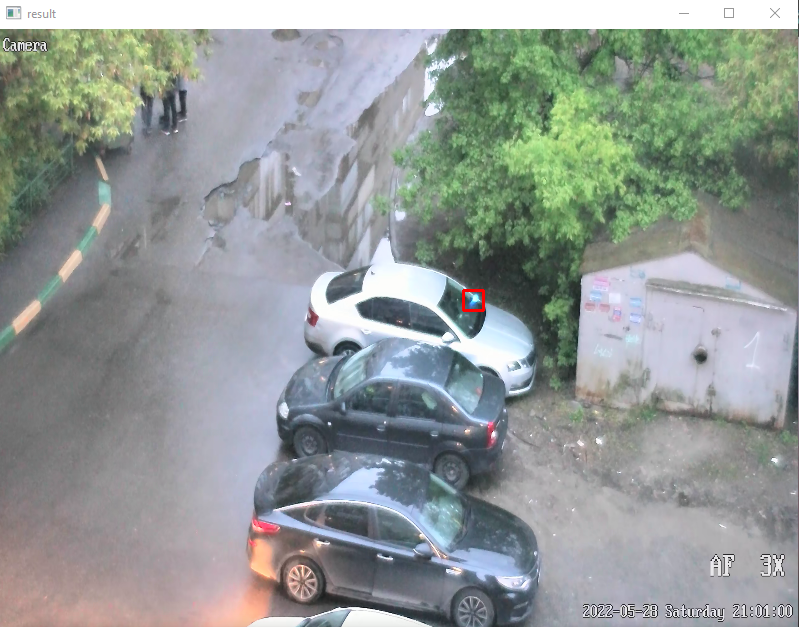


Рисунок 6 – Детектирование светодиода в ночное время

Также проверим работу программы в других условиях, в вечернее время (Рисунок 7).

Рисунок 7 – Детектирование светодиода в вечернее время

Однако из-за того, что мы находим оттенки синего на видеопотоке так же могут помечаться прохожие и машины (Рисунок 8).

Рисунок 8 – Ложное срабатывание системы

Для решения этой проблемы в дальнейшем будет использоваться ИК-диод, так как он менее распространён, чем синие светодиоды, и у него можно повысить мощность излучения для более лучшего обнаружения системой в любое время суток. А также будет реализован алгоритм для выявления определённой назначенной последовательности мигания ИК-диода.

# **Код**

Find\_Color.py

import cv2

import numpy as np

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

def nothing(\*arg):

pass

cv2.namedWindow( "result" ) # создаем главное окно

cv2.namedWindow( "settings" ) # создаем окно настроек

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

# создаем 6 бегунков для настройки начального и конечного цвета фильтра

cv2.createTrackbar('h1', 'settings', 0, 255, nothing)

cv2.createTrackbar('s1', 'settings', 0, 255, nothing)

cv2.createTrackbar('v1', 'settings', 0, 255, nothing)

cv2.createTrackbar('h2', 'settings', 255, 255, nothing)

cv2.createTrackbar('s2', 'settings', 255, 255, nothing)

cv2.createTrackbar('v2', 'settings', 255, 255, nothing)

crange = [0,0,0, 0,0,0]

try:

while True:

flag, img = cap.read()

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

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

# считываем значения бегунков

h1 = cv2.getTrackbarPos('h1', 'settings')

s1 = cv2.getTrackbarPos('s1', 'settings')

v1 = cv2.getTrackbarPos('v1', 'settings')

h2 = cv2.getTrackbarPos('h2', 'settings')

s2 = cv2.getTrackbarPos('s2', 'settings')

v2 = cv2.getTrackbarPos('v2', 'settings')

# формируем начальный и конечный цвет фильтра

h\_min = np.array((h1, s1, v1), np.uint8)

h\_max = np.array((h2, s2, v2), np.uint8)

# накладываем фильтр на кадр в модели HSV

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

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

cv2.imshow('Origin', img)

cv2.imshow('result', thresh)

ch = cv2.waitKey(1)

if ch == 27:

break

except KeyboardInterrupt:

print(' Exit pressed Ctrl+C')

cv2.destroyAllWindows()

Find\_Diode.py

import cv2

import numpy as np

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

def callback(\*arg):

print (arg)

cv2.namedWindow( "result" )

cap = cv2.VideoCapture('3.mp4')

# HSV фильтр для синих объектов

while True:

flag, img = cap.read()

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

# преобразуем RGB картинку в HSV модель

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

hsv\_min = np.array((0, 0, 236), np.uint8)

hsv\_max = np.array((158, 255, 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 10 пикселей

if dArea > 10:

x = int(dM10 / dArea)

y = int(dM01 / dArea)

cv2.rectangle(img, (x-10, y-10), (x+10, y+10), (0,0,255), 2)

cv2.imshow('result', img)

cv2.imshow('obj', thresh)

ch = cv2.waitKey(1)

if ch == 27:

break

cap.release()

cv2.destroyAllWindows()

# **Вывод**

За время прохождения производственной научно-исследовательской практики был изучить принцип работы компьютерного зрения, была изучена библиотека OpenCV.

В ходе работы была разработана система, которая находит на видеоматериале с видеокамеры светодиод сигнализации машины, путём выделения нужного цвета светодиода на видеопотоке и детектирования его.

# **Список литературы**

1. Компьютерное зрение [Электронный ресурс] – URL: http://neerc.ifmo.ru/wiki/index.php?title=Компьютерное\_зрение. (Дата обращения: 20.02.2022)
2. OpenCV [Электронный ресурс] – URL: https://blog.skillfactory.ru/glossary/opencv/. (Дата обращения: 12.03.2022)