**ПРИЛОЖЕНИЕ А**

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

«БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

# КАФЕДРА ИНТЕЛЛЕКТУАЛЬНЫХ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

РАЗРАБОТКА ПРОСТОЙ МОДЕЛИ РАСПОЗНАВАНИЯ

ГОЛОСОВЫХ КОМАНД

**ТЕКСТ ПРОГРАММЫ**

КП.ПО4.190348-04 12 00

## Листов 5

|  |  |
| --- | --- |
| Руководитель | Кочурко П.А |
| Выполнил | Синяк Д.А |
| Консультант |  |
| по ЕСПД | Кочурко П.А |
|  |  |

2021

**СОДЕРЖАНИЕ**

def p\_centro – функция для нахождения центра отслежеваемого объекта.

cv2.VideoCapture – предназначенно для загрузки видеопотока.

cap.read() – предназначенно для прочтения видеопотока.

cv2.cvtColor – предназначенно для преобразования цветового пространства.

cv2.GaussianBlur – предназначенно для сглаживания видеопотока.

cv2.line – накладывает на видеопоток горизонтальную линию.

cv2.rectangle – захватывает отслежеваемый объект в зелёный прямоугольник.

cv2.circle – накладывает в центр отслежеваемого объекта красную точку.

cv2.putText – выводит на экран текст.

cv2.imshow – выводит на экран фрейм с которым происходит работа.

**Код программы:**

import cv2

import numpy as np

from time import sleep

l\_min = 80

# минимальная ширина прямоугольника

a\_min = 80

# минимальная высота прямоугольника

offset = 6

# допустимая ошибка между пикселями

pos\_l = 650

# доложение линии счета

delay = 60

# видео FPS(Количество кадров в секунду)

detec = []

car = 0

def p\_centro(x, y, w, h):

x1 = int(w / 2)

y1 = int(h / 2)

cx = x + x1

cy = y + y1

return cx, cy

# указывает на середину

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

subtracao = cv2.bgsegm.createBackgroundSubtractorMOG()

# cоздать фоновый вычитатель

while True:

# если данные верны, выполнить цикл

ret, frame1 = cap.read()

# прочитать видео

t = float(1 / delay)

# kоличество кадров в секунду (приравнять к переменной)

sleep(t)

# время сна

grey = cv2.cvtColor(frame1, cv2.COLOR\_BGR2GRAY)

# преобразования цветового пространства

blur = cv2.GaussianBlur(grey, (3, 3), 5)

# размытие изображений

img\_sub = subtracao.apply(blur)

dilat = cv2.dilate(img\_sub, np.ones((5, 5)))

kernel = cv2.getStructuringElement(cv2.MORPH\_ELLIPSE, (5, 5))

dilatada = cv2.morphologyEx(dilat, cv2.MORPH\_CLOSE, kernel)

dilatada = cv2.morphologyEx(dilatada, cv2.MORPH\_CLOSE, kernel)

contor, h = cv2.findContours(dilatada, cv2.RETR\_TREE, cv2.CHAIN\_APPROX\_SIMPLE)

cv2.line(frame1, (25, pos\_l), (1200, pos\_l), (255, 127, 0), 3)

for (i, c) in enumerate(contor):

(x, y, w, h) = cv2.boundingRect(c)

validate = (w >= l\_min) and (h >= a\_min)

if not validate:

continue

# подтверждать линии

cv2.rectangle(frame1, (x, y), (x + w, y + h), (0, 255, 0), 2)

centro = p\_centro(x, y, w, h)

detec.append(centro)

cv2.circle(frame1, centro, 4, (0, 0, 255), 10)

# построить круги в списке координат центра

for (x, y) in detec:

if y < (pos\_l + offset) and y > (pos\_l - offset):

car += 1

cv2.line(frame1, (25, pos\_l), (1200, pos\_l), (0, 127, 255), 3)

detec.remove((x, y))

print("car\_detected : " + str(car))

# этот цикл вычисляет количество машин, которые пересекают введенную нами линию

cv2.putText(frame1, "AUTO COUNT : " + str(car), (450, 70), cv2.FONT\_HERSHEY\_SIMPLEX, 2, (0, 0, 255), 5)

# печатать количество car

cv2.imshow("Video Original", frame1)

if cv2.waitKey(1) == 7:

break

# if 7 окно закрывается

cv2.destroyAllWindows()

cap.release()