



Министерство науки и высшего образования Российской
Федерации
Федеральное государственное бюджетное образовательное
учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ
КАФЕДРА КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ
НАПРАВЛЕНИЕ ПОДГОТОВКИ 09.03.01 Компьютерные системы и сети.

О Т Ч Е Т П О П Р О И З В О Д С Т В Е Н Н О Й
П Р А К Т И К Е

Студент Медведев Александр Евгеньевич

Группа ИУ6-42

Тип практики Производственная

Название предприятия RTSoft

Студент _____ Медведев АЕ
(Подпись, дата) (И.О. Фамилия)

Руководитель практики _____
(Подпись, дата) (И.О. Фамилия)

Оценка _____

СОДЕРЖАНИЕ

ВВЕДЕНИЕ.....	3
1. Udev правило	4
2. Создание драйвера	6
3. Изучение docker	10
4. Поиск контрастных объектов с помощью библиотеки OpenCV.....	11
5. Алгоритм сглаживания	15
6. Изучение mqtt.....	19
7. Изучение kafka	24
8. Индивидуальный проект – Анализ дорожной разметки.....	25
Заключение	29
Список литературы	29

Введение

В период с 5 по 23 июля в компании РТСофт по направлению «Разработка системного программного обеспечения для интеллектуальных Embedded IoT устройств» проходила летняя практика. Программа была разбита на две части: курс лекций с домашними заданиями и индивидуальный проект.

Целью практики было получение теоритических знаний и практических навыков о технологиях, которые используется в современной разработке.

В ходе прохождения практики были поставлены следующие задачи: изучить работу ОС Linux, драйверов, docker, алгаритмам компьютерного зрения и работы библиотеки OpenCV. А также разработать алгоритм отпределения дорожной разметки из видео потока.

1. Udev правило

Udev (userspace /dev) — это подсистема Linux для динамического обнаружения устройств и управления ими.

Необходимо написать правило, которое копировало логи системы на заданную вылешку при её подключении к компьютеру.

Создадим файл **expansion-usb.rules** и запишем следующее правило:
SUBSYSTEM=="block", ACTION=="add", ENV{DEVTYPE}=="disk",
ATTRS{idVendor}=="abcd", ATTRS{idProduct}=="1234",
RUN+="/usr/bin/expansion-usb.sh"

Это правило описывает, что при подключении флешки с id=abcd:1234 нужно запустить скрипт **expansion-usb.sh** в директории **/usr/bin/**. Просмотр атрибутов возможен через команду **udevinfo**.

Файл содержит следующий код:

```
#!/bin/bash                                - исполняемый файл
sudo mkdir /media/temp                     - создание директории для подключения usb
sudo mount /dev/sdb1 /media/temp           - монтирование диска к этой директории
dmesg > /media/temp/dmsg_logs.txt          - копирование логов на диск
sudo umount /media/temp                   - отмонтирование диска от системы
```

Чтобы прило заработало, надо перенести файл **expansion-usb.rules** в директорию **/etc/udev/rules.d/**

На рисунке 1 показано состояние флеш-накопителя до применения правила, на рисунке 2 — после. На рисунок 3 изображена характеристика флеш-накопителя.

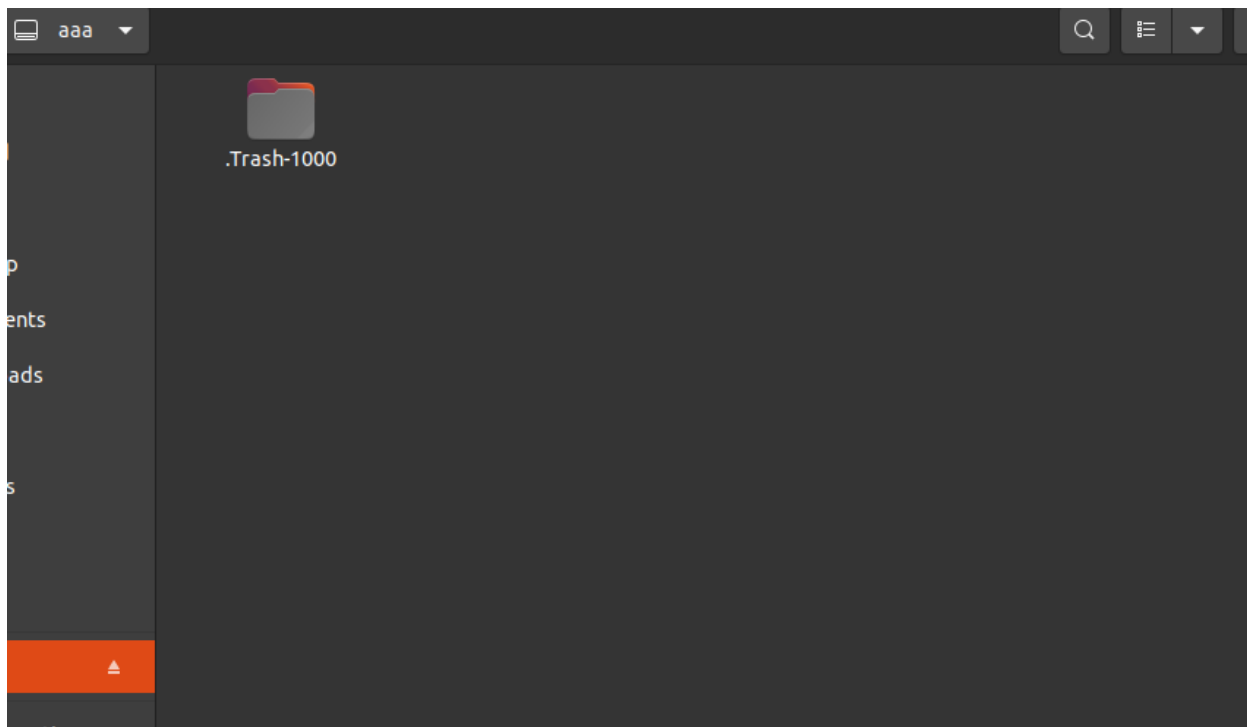


Рисунок 1 — флеш-накопитель до работы правила

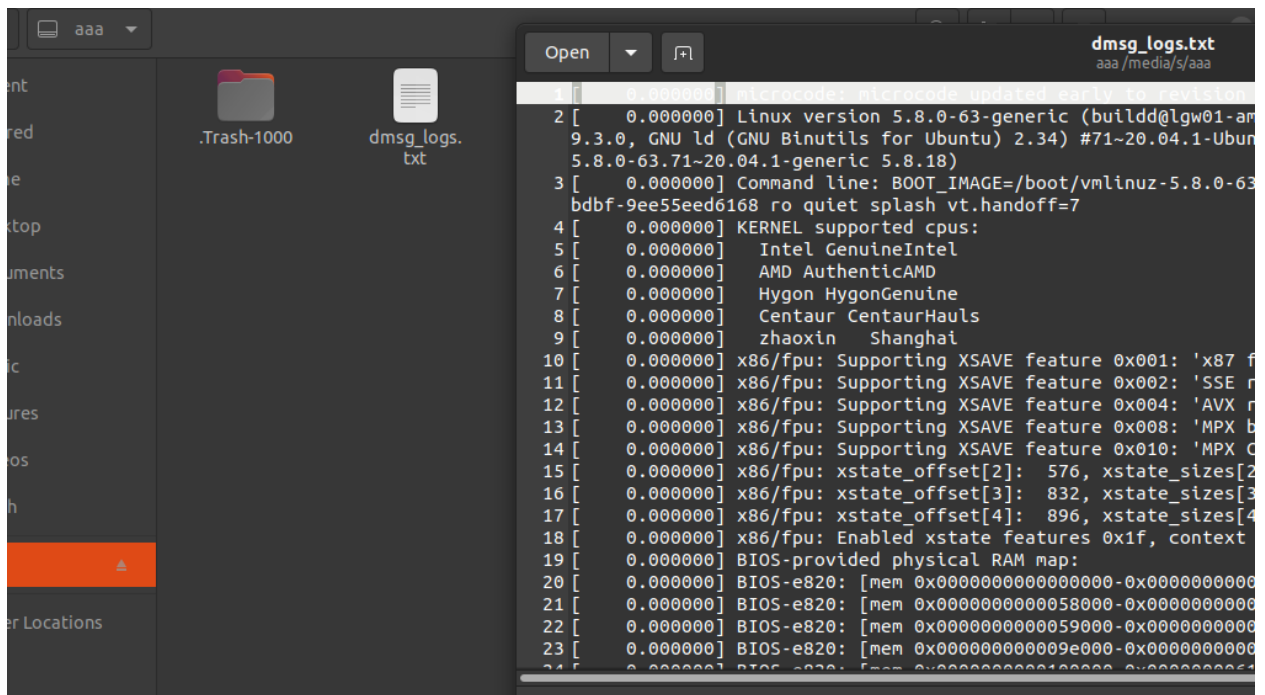


Рисунок 2 — флеш-накопитель после работы правила

```
s@s-Aspire-A715-71G ~/Desktop/RTSoft/task1 master udevadm info -a -n sdb1

udevadm info starts with the device specified by the devpath and then
walks up the chain of parent devices. It prints for every device
found, all possible attributes in the udev rules key format.
A rule to match, can be composed by the attributes of the device
and the attributes from one single parent device.

looking at device '/devices/pci0000:00/0000:00:14.0/usb1/1-5/1-5:1.0/host3/target3:0:0/
KERNEL=="sdb1"
SUBSYSTEM=="block"
DRIVER==" "
ATTR{alignment_offset}=="0"
ATTR{inflight}=="          0          0"
ATTR{stat}=="          229          30307          34985          4945          27          2          334          103
0          0          0          0          0          0"
ATTR{start}=="2048"
ATTR{ro}=="0"
ATTR{size}=="31125504"
ATTR{discard_alignment}=="0"
ATTR{partition}=="1"

looking at parent device '/devices/pci0000:00/0000:00:14.0/usb1/1-5/1-5:1.0/host3/targe
KERNELS=="sdb"
SUBSYSTEMS=="block"
DRIVERS==" "
ATTRS{removable}=="1"
```

Рисунок 3 — информация о флеш-накопителе

2. Создание драйвера

Драйвер — программ для связывания двух разных компонентов. К примеру, компьютера и принтера.

Задача написать оболочку драйвера, которая изменяет внутреннюю переменную на 1 при считывании 1 байта информации.

Файл test.c — файл оболочки драйвера.

```
#include <linux/kernel.h>
```

```
#include <linux/module.h>
```

```
#include <linux/init.h>
```

```
#include <linux/fs.h>
```

```
#include <asm/uaccess.h>
```

```
MODULE_LICENSE( "GPL" );
```

```
MODULE_SUPPORTED_DEVICE( "test" );
```

```
#define SUCCESS 0
```

```
#define DEVICE_NAME "test"
```

```
static int device_open( struct inode *, struct file * );
```

```
static int device_release( struct inode *, struct file * );
```

```
static ssize_t device_read( struct file *, char *, size_t, loff_t * );
```

```
static ssize_t device_write( struct file *, const char *, size_t, loff_t * );
```

```
static int major_number;
```

```
static int is_device_open = 0;
```

```
static int magic_number = 50;    // наше число.
```

```
static char text[ 5 ] = "test\n";
```

```
static char* text_ptr = text;
```

```
static struct file_operations fops =
```

```
{
```

```
    .read = device_read,
```

```
    .write = device_write,
```

```
    .open = device_open,
```

```
    .release = device_release
```

```
};
```

```
static int __init test_init( void )
```

```
{
```

```
    printk( KERN_ALERT "TEST driver loaded!\n" );
```

```
    major_number = register_chrdev( 0, DEVICE_NAME, &fops );
```

```
    if ( major_number < 0 )
```

```
{
```

```

    printk( "Registering the character device failed with %d\n", major_number );
    return major_number;
}

printk( "Test module is loaded!\n" );

    printk( "Please, create a dev file with 'mknod /dev/test c %d 0'.\n",
major_number );

    return SUCCESS;
}

static void __exit test_exit( void )
{
    unregister_chrdev( major_number, DEVICE_NAME );

    printk( KERN_ALERT "Test module is unloaded!\n" );
}

module_init( test_init );
module_exit( test_exit );

static int device_open( struct inode *inode, struct file *file )
{
    text_ptr = text;

    if ( is_device_open )
        return -EBUSY;

    is_device_open++;

    return SUCCESS;
}

static int device_release( struct inode *inode, struct file *file )
{
    is_device_open--;
    return SUCCESS;
}

static ssize_t device_write( struct file *filp, const char *buff, size_t len, loff_t * off
)
{
    printk( "Sorry, this operation isn't supported.\n" );
    return -EINVAL;
}

```



```

}

static ssize_t device_read( struct file *filp,
                           char *buffer,
                           size_t length,
                           loff_t * offset )
{
    int byte_read = 0;

    if ( *text_ptr == 0 )
        return 0;

    while ( length && *text_ptr )
    {
        put_user( magic_number++ , buffer );
        length--;
        byte_read++;
    }
    return byte_read;
}

```

Makefile – файл сборки драйвера.

obj-m += test.o

all:

make -C /lib/modules/\$(shell uname -r)/build M=\$(PWD) modules

clean:

make -C /lib/modules/\$(shell uname -r)/build M=\$(PWD) clean

main.c – файл проверки работоспособности драйвера.

```
#include <sys/types.h>
```

```
#include <sys/stat.h>
```

```
#include <fcntl.h>
```

```
#include <stdio.h>
```

```
#include <stdlib.h>
```

```
#include <unistd.h>
```

```
int main()
```

```
{
```

```
    int fd;
```

```
    char buf[100];
```

```
    fd = open("/dev/test",O_RDONLY);
```

```
    if (!fd)
```

```
        printf("Error open!!!\n");
```

```

if (!read(fd,buf,1))
    printf("Error read!!!\n");
buf[20]=0;
printf("Input: >>> %d <<<\n",*buf);
close(fd);

return 0;
}

```

На рисунке 4 представлен результат.

```

s@s-Aspire-A715-71G ~/Desktop/RTSoft/task2 master ± ./a.out
Input: >>> 50 <<<
s@s-Aspire-A715-71G ~/Desktop/RTSoft/task2 master ± ./a.out
Input: >>> 51 <<<
s@s-Aspire-A715-71G ~/Desktop/RTSoft/task2 master ± ./a.out
Input: >>> 52 <<<
s@s-Aspire-A715-71G ~/Desktop/RTSoft/task2 master ± ./a.out
Input: >>> 53 <<<
s@s-Aspire-A715-71G ~/Desktop/RTSoft/task2 master ±

```

Рисунок 4 — работа драйвера.

3. Изучение docker

Docker — программное обеспечение для автоматизации развёртывания и управления приложениями в средах с поддержкой контейнеризации, контейнеризатор приложений. Позволяет «упаковать» приложение со всем его окружением и зависимостями в контейнер, который может быть развёрнут на любой Linux - системе с поддержкой в ядре, а также предоставляет набор команд для управления этими контейнерами.

Задача написать докерфайл для установки браузера mozilla firefox и запуск его с проброшенными иксами. Результат работы представлен на рисунке 5.

Dockerfile

```
FROM ubuntu:20.04
```

```
MAINTAINER example
```

```
RUN apt-get -yqq update
```

```
RUN apt-get install -y firefox
```

```
CMD firefox
```

Запуск

```
sudo xhost +local:root
```

```
sudo docker build -t new_firefox_docker .
```

```
sudo docker run -it --net=host --env="DISPLAY" new_firefox_docker
```

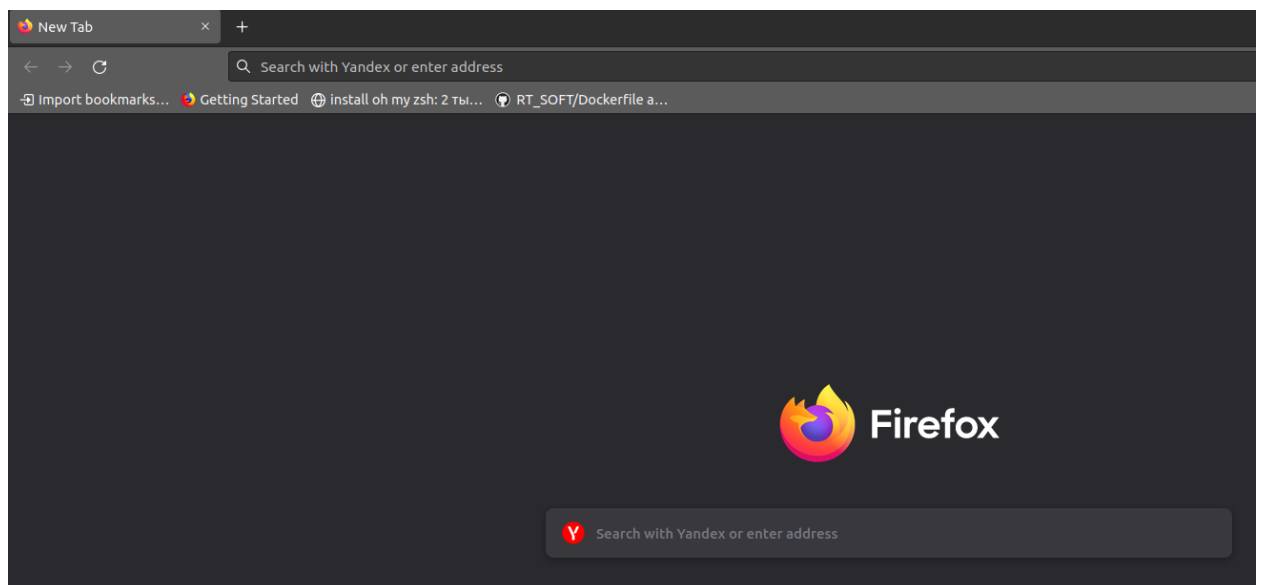


Рисунок 5 — запуск firefox через докер.

4. Поиск контрастных объектов с помощью библиотеки OpenCV

Для поиска контрастных объектов используется библиотека openCV. CvtColor — переводит изображение в удобный формат. InRange — работает с контрастностью и глубиной цвета, findContours — ищет контуры.

Результат можно наблюдать на рисунке 6.

Файл main.cpp

```
#include <iostream>
```

```
#include <vector>
```

```
#include <algorithm>
```

```
#include <opencv2/opencv.hpp>
```

```
#include <opencv2/highgui.hpp>
```

```
#include <math.h>
```

```
void recogniseStickersByThreshold(cv::Mat image,
```

```
std::vector<std::vector<cv::Point>> &stickers) {
```

```
    cv::Mat image_hsv;
```

```
    std::vector< std::vector<cv::Point> > contours;
```

```
    cv::cvtColor(image, image_hsv, cv::COLOR_BGR2HSV );
```

```
    cv::Mat tmp_img(image.size(), CV_8U);
```

```
    cv::inRange(image_hsv, cv::Scalar(100, 100, 100), cv::Scalar(255, 255, 255),  
tmp_img);
```

```
    cv::dilate(tmp_img,tmp_img,cv::Mat(), cv::Point(-1,-1), 3);
```

```
    cv::erode(tmp_img,tmp_img,cv::Mat(), cv::Point(-1,-1), 1);
```

```
    cv::findContours(tmp_img, stickers, cv::RETR_EXTERNAL,  
cv::CHAIN_APPROX_NONE);
```

```
    for (uint i = 0; i<contours.size(); i++) {
```

```
        cv::Mat sticker;
```

```
        cv::Rect rect=cv::boundingRect(contours[i]);
```

```
        image(rect).copyTo(sticker);
```

```
        cv::rectangle(image, rect, cv::Scalar(0, 250, 0), 2);
```

```
        stickers.push_back(sticker);
```

```
    }
```

```
}
```

```
bool cmpPointX(const cv::Point &a, const cv::Point &b) {
```

```
    return a.x < b.x;
```

```
}
```

```
bool cmpPointY(const cv::Point &a, const cv::Point &b) {
```

```
    return a.y < b.y;
```

```
}
```

```

int main() {
    using namespace cv;
    using namespace std;

    cout << "start" << endl;
    VideoCapture cap("./project/temp.mp4");
    if ( !cap.isOpened() ) return -1;
    namedWindow("MyVideo",cv::WindowFlags::WINDOW_AUTOSIZE);

    std::vector<std::vector<cv::Point>> stickers;
    while(1) {
        Mat frame;
        bool bSuccess = cap.read(frame);
        if (!bSuccess) {
            cout << "Cannot read the frame from video file" << endl;
            break;
        }

        recogniseStickersByThreshold(frame, stickers);
        for (auto st : stickers)
        {
            cv::Point sticker1;
            sticker1.x = (*min_element(st.begin(), st.end(), cmpPointX)).x;
            sticker1.y = (*min_element(st.begin(), st.end(), cmpPointY)).y;
            cv::Point sticker2;
            sticker2.x = (*max_element(st.begin(), st.end(), cmpPointX)).x;
            sticker2.y = (*max_element(st.begin(), st.end(), cmpPointY)).y;
            cv::rectangle(frame, Rect(sticker1, sticker2),cv::Scalar(0,250,0),2);
        }

        imshow("MyVideo", frame);
        if(waitKey(30) == 27) {
            break;
        }
    }
    return 0;
}

```

Упаковочный файл — Dockerfile

FROM ubuntu:18.04

RUN apt update

RUN apt install python3-opencv -y && apt-get install -y wget

RUN apt install -y build-essential cmake git pkg-config libgtk-3-dev \
libavcodec-dev libavformat-dev libswscale-dev libv4l-dev \
libxvidcore-dev libx264-dev libjpeg-dev libpng-dev libtiff-dev \

```
gfortran openexr libatlas-base-dev python3-dev python3-numpy \
libtbb2 libtbb-dev libdc1394-22-dev
```

```
RUN mkdir ~/opencv_build && cd ~/opencv_build && git clone
https://github.com/opencv/opencv.git
```

```
RUN git clone https://github.com/opencv/opencv_contrib.git
```

```
RUN cd ~/opencv_build/opencv && mkdir build && cd build \
&& cmake -DCMAKE_CXX_FLAGS=-std=c++11 \
-DCMAKE_BUILD_TYPE=RELEASE \
-DCMAKE_INSTALL_PREFIX=/usr/local \
-DBUILD_EXAMPLES=OFF \
-DBUILD_DOCS=OFF \
-DBUILD_PERF_TESTS=OFF \
-DBUILD_TESTS=OFF \
-DINSTALL_C_EXAMPLES=OFF \
-DENABLE_PRECOMPILED_HEADERS=OFF \
-DWITH_OPENMP=ON \
-DWITH_V4L=ON \
-DWITH_TBB=ON \
-DWITH_OPENGL=ON \
-DWITH_JPEG=ON \
-DWITH_FFMPEG=ON \
-DWITH_GSTREAMER=ON \
-DWITH_OPENCL=ON \
-DWITH_GPHOTO2=ON \
-DWITH_LIBV4L=ON \
-DINSTALL_PYTHON_EXAMPLES=ON \
-DBUILD_SHARED_LIBS=ON \
-DENABLE_CXX11=ON ..
```

```
RUN cd ~/opencv_build/opencv/build \
&& make -j8 \
&& make install
```

```
RUN cd / && mkdir project
```

```
COPY doc /project
```

```
RUN cd project && ls && cmake CMakeLists.txt && make
CMD ./project/main
```

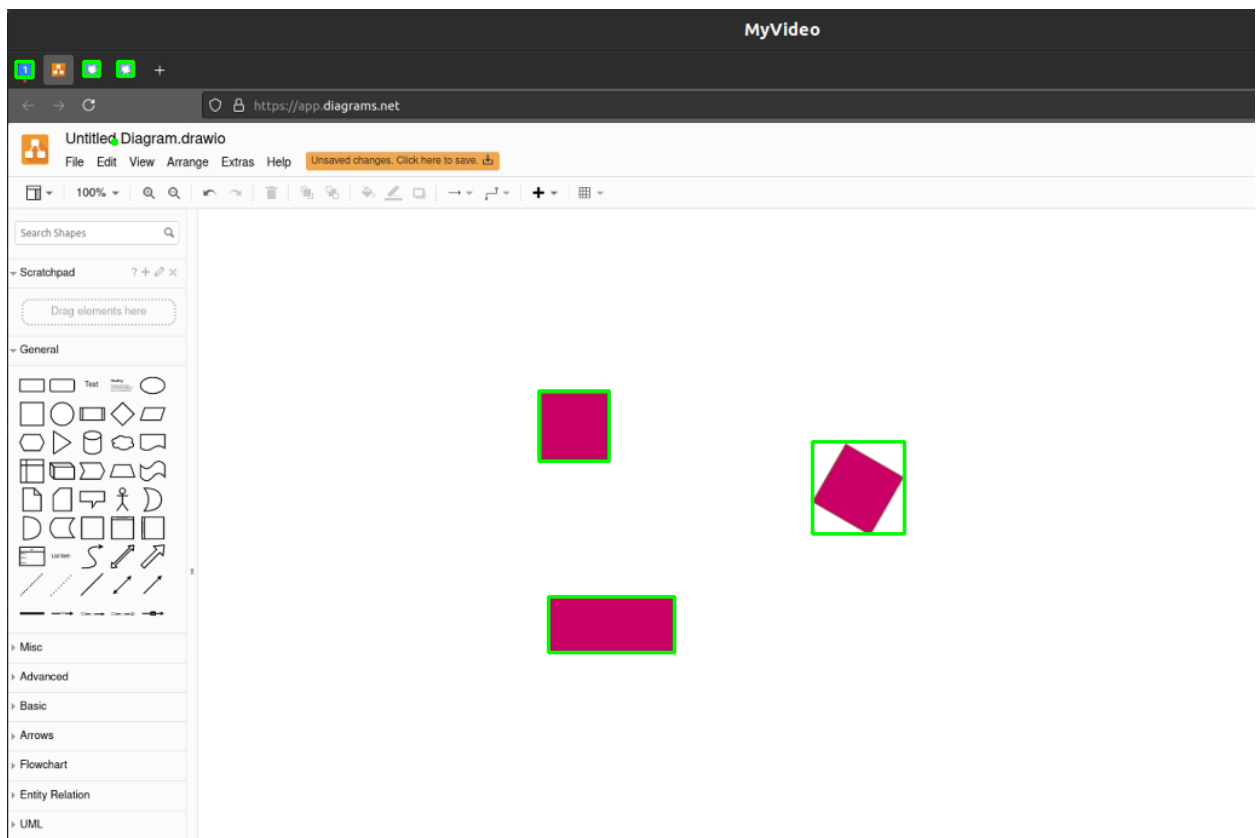


Рисунок 6 — выделение контрастных объектов.

5. Алгоритм сглаживания

В качестве алгоритма сглаживания был изучен метод среднего скользящего. Зелёный прямоугольник — выделенный объект, Фиолетовый прямоугольник — сглаженный объект. Результат представлен на рисунке 7.

Файл main.cpp

```
#include <iostream>
```

```
#include <vector>
```

```
#include <algorithm>
```

```
#include <opencv2/opencv.hpp>
```

```
#include <opencv2/highgui.hpp>
```

```
#include <math.h>
```

```
void recogniseStickersByThreshold(cv::Mat image,
```

```
std::vector<std::vector<cv::Point>> &stickers) {
```

```
    cv::Mat image_hsv;
```

```
    std::vector< std::vector<cv::Point> > contours;
```

```
    cv::cvtColor(image, image_hsv, cv::COLOR_BGR2HSV );
```

```
    cv::Mat tmp_img(image.size(), CV_8U);
```

```
    cv::inRange(image_hsv, cv::Scalar(100, 100, 100), cv::Scalar(255, 255, 255),  
tmp_img);
```

```
    cv::dilate(tmp_img,tmp_img,cv::Mat(), cv::Point(-1,-1), 3);
```

```
    cv::erode(tmp_img,tmp_img,cv::Mat(), cv::Point(-1,-1), 1);
```

```
    cv::findContours(tmp_img, stickers, cv::RETR_EXTERNAL,  
cv::CHAIN_APPROX_NONE);
```

```
    for (uint i = 0; i<contours.size(); i++) {
```

```
        cv::Mat sticker;
```

```
        cv::Rect rect=cv::boundingRect(contours[i]);
```

```
        image(rect).copyTo(sticker);
```

```
        cv::rectangle(image, rect, cv::Scalar(0, 250, 0), 2);
```

```
        stickers.push_back(sticker);
```

```
    }
```

```
}
```

```
bool cmpPointX(const cv::Point &a, const cv::Point &b) {
```

```
    return a.x < b.x;
```

```
}
```

```
bool cmpPointY(const cv::Point &a, const cv::Point &b) {
```

```
    return a.y < b.y;
```

```
}
```

```
cv::Point newGain(cv::Point oldGain, cv::Point measurement, cv::Point &target) {
```

```
    target += oldGain;
```



```

    std::cout << target << "\n";
    return oldGain + (1/3) * (measurement - target);
}

cv::Point estimate(cv::Point prediction, cv::Point measurement) {
    return 0.6 * prediction + 0.4 * measurement;
}

int main() {
    using namespace cv;
    using namespace std;

    cout << "start" << endl;
    VideoCapture cap("./project/temp.mp4");
    if ( !cap.isOpened() ) return -1;
    namedWindow("MyVideo",cv::WindowFlags::WINDOW_AUTOSIZE);

    std::vector<std::vector<cv::Point>> stickers;
    cv::Point gain1(1, 1);
    cv::Point gain2(1, 1);
    cv::Point prediction1(1, 1);
    cv::Point prediction2(1, 1);
    bool start = true;
    while(1) {
        Mat frame;
        bool bSuccess = cap.read(frame);
        if (!bSuccess) {
            cout << "Cannot read the frame from video file" << endl;
            break;
        }

        recogniseStickersByThreshold(frame, stickers);

        if (start)
        {
            auto st = stickers.front();
            cv::Point sticker1;
            sticker1.x = (*min_element(st.begin(), st.end(), cmpPointX)).x;
            sticker1.y = (*min_element(st.begin(), st.end(), cmpPointY)).y;
            prediction1 = sticker1;
            cv::Point sticker2;
            sticker2.x = (*max_element(st.begin(), st.end(), cmpPointX)).x;
            sticker2.y = (*max_element(st.begin(), st.end(), cmpPointY)).y;
            prediction2 = sticker2;
            start = false;
        }
    }
}

```

```

    }

    for (auto st : stickers)
    {
        cv::Point sticker1;
        sticker1.x = (*min_element(st.begin(), st.end(), cmpPointX)).x;
        sticker1.y = (*min_element(st.begin(), st.end(), cmpPointY)).y;
        cv::Point sticker2;
        sticker2.x = (*max_element(st.begin(), st.end(), cmpPointX)).x;
        sticker2.y = (*max_element(st.begin(), st.end(), cmpPointY)).y;
        cv::rectangle(frame, Rect(sticker1, sticker2), cv::Scalar(0,250,0),2);

        gain1 = newGain(gain1, sticker1, prediction1);
        gain2 = newGain(gain2, sticker2, prediction2);

        prediction1 = estimate(prediction1, sticker1);
        prediction2 = estimate(prediction2, sticker2);
        cv::rectangle(frame, Rect(prediction1 + gain1, prediction2
+gain2), cv::Scalar(250,0,0),2);
        break;
    }

    imshow("MyVideo", frame);
    if(waitKey(30) == 27) {
        break;
    }
}
return 0;
}

```

Упаковочный файл — Dockerfile

FROM ubuntu:18.04

RUN apt update

RUN apt install python3-opencv -y && apt-get install -y wget

RUN apt install -y build-essential cmake git pkg-config libgtk-3-dev \
 libavcodec-dev libavformat-dev libswscale-dev libv4l-dev \
 libxvidcore-dev libx264-dev libjpeg-dev libpng-dev libtiff-dev \
 gfortran openexr libatlas-base-dev python3-dev python3-numpy \
 libtbb2 libtbb-dev libdc1394-22-dev

RUN mkdir ~/opencv_build && cd ~/opencv_build && git clone
<https://github.com/opencv/opencv.git>

RUN git clone https://github.com/opencv/opencv_contrib.git

RUN cd ~/opencv_build/opencv && mkdir build && cd build \
 && cmake -DCMAKE_CXX_FLAGS=-std=c++11 \
 -DCMAKE_BUILD_TYPE=RELEASE \
 -DCMAKE_INSTALL_PREFIX=/usr/local \
 -DBUILD_EXAMPLES=OFF \

```

-DBUILD_DOCS=OFF \
-DBUILD_PERF_TESTS=OFF \
-DBUILD_TESTS=OFF \
-DINSTALL_C_EXAMPLES=OFF \
-DENABLE_PRECOMPILED_HEADERS=OFF \
-DWITH_OPENMP=ON \
-DWITH_V4L=ON \
-DWITH_TBB=ON \
-DWITH_OPENGL=ON \
-DWITH_JPEG=ON \
-DWITH_FFMPEG=ON \
-DWITH_GSTREAMER=ON \
-DWITH_OPENCL=ON \
-DWITH_GPHOTO2=ON \
-DWITH_LIBV4L=ON \
-DINSTALL_PYTHON_EXAMPLES=ON \
-DBUILD_SHARED_LIBS=ON \
-DENABLE_CXX11=ON ..
RUN cd ~/opencv_build/opencv/build \
  && make -j8 \
  && make install
RUN cd / && mkdir project
COPY doc /project
RUN cd project && ls && cmake CMakeLists.txt && make
CMD ./project/main

```

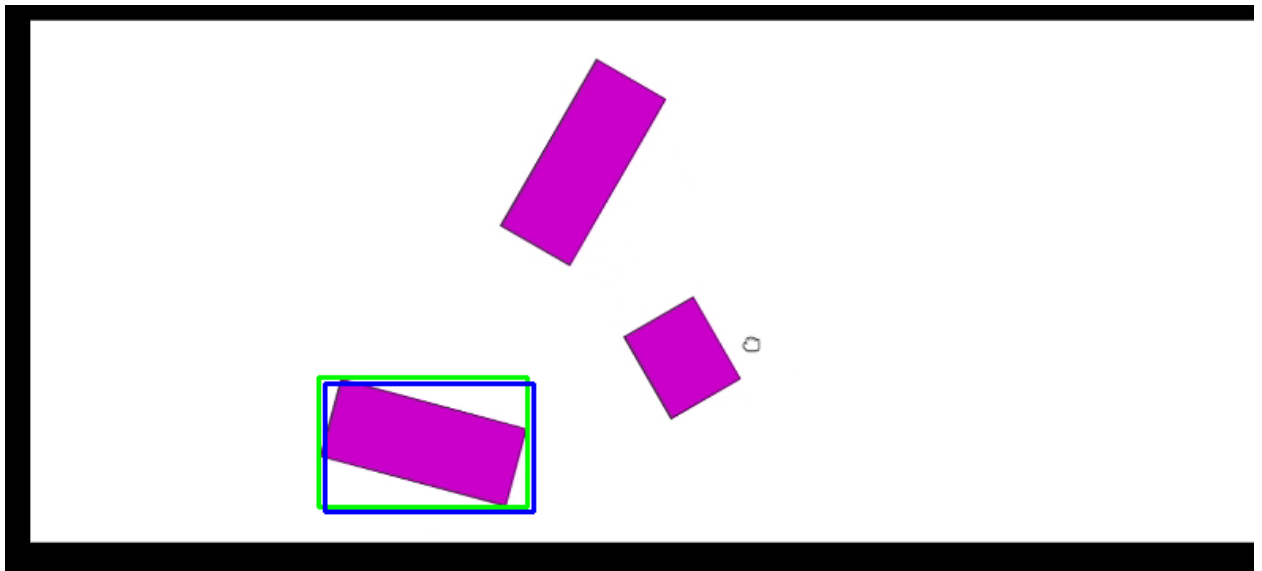


Рисунок 7 — работа сглаживающего алгоритма.

6. Изучение mqtt

Задача изучить работы mqtt, записать координаты контрастного объекта в формат json и передать пакетом четез локальную сеть. Все файлы собрать в докер. Результат представлен на рисунке 8.

```
#include <signal.h>
#include <stdio.h>
#include <stdint.h>
#include <string.h>
#define mqtt_host "localhost"
#define mqtt_port 1880
#include <mosquitto.h>

#include <iostream>
#include <vector>
#include <algorithm>
#include <opencv2/opencv.hpp>
#include <opencv2/highgui.hpp>
#include <math.h>
#include <nlohmann/json.hpp>
using nlohmann::json;

void recogniseStickersByThreshold(cv::Mat image,
std::vector<std::vector<cv::Point>> &stickers) {
    cv::Mat image_hsv;
    std::vector< std::vector<cv::Point> > contours;
    cv::cvtColor(image, image_hsv, cv::COLOR_BGR2HSV );
    cv::Mat tmp_img(image.size(), CV_8U);
    cv::inRange(image_hsv, cv::Scalar(100, 100, 100), cv::Scalar(255, 255, 255),
tmp_img);
    cv::dilate(tmp_img,tmp_img,cv::Mat(), cv::Point(-1,-1), 3);
    cv::erode(tmp_img,tmp_img,cv::Mat(), cv::Point(-1,-1), 1);
    cv::findContours(tmp_img, stickers, cv::RETR_EXTERNAL,
cv::CHAIN_APPROX_NONE);
    for (uint i = 0; i<contours.size(); i++) {
        cv::Mat sticker;
        cv::Rect rect=cv::boundingRect(contours[i]);
        image(rect).copyTo(sticker);
        cv::rectangle(image, rect, cv::Scalar(0, 250, 0), 2);
        stickers.push_back(sticker);
    }
}

bool cmpPointX(const cv::Point &a, const cv::Point &b) {
    return a.x < b.x;
}
```

```

bool cmpPointY(const cv::Point &a, const cv::Point &b) {
    return a.y < b.y;
}

struct Box {
    int x1, y1;
    int x2, y2;
    std::string name;
};

void to_json(json& j, const Box& p)
{
    j = json{
        { "vector1", {
            { "x", p.x1 },
            { "y", p.y1 }
        }
        },
        { "vector2", {
            { "x", p.x2 },
            { "y", p.y2 }
        }
        },
        { "name", p.name }
    };
}

using namespace cv;
using namespace std;

int main() {

    int rc;
    struct mosquitto *mosq;

    mosq = mosquitto_new("pubisher-test", true, NULL);

    rc = mosquitto_connect(mosq, mqtt_host, mqtt_port, 60);
    if (rc != 0)
    {
        printf("Client could not connect to brocker! Error Code: %d\n", rc);
        mosquitto_destroy(mosq);
        return -1;
    }
}

```

```

printf("We are now connected to the broker!\n");

cout << "start" << endl;
VideoCapture cap("./temp.mp4");
if ( !cap.isOpened() ) return -1;
namedWindow("MyVideo",cv::WindowFlags::WINDOW_AUTOSIZE);
cout << "video" << endl;
std::string name = "box";
std::vector<std::vector<cv::Point>> stickers;
while(1) {
    Mat frame;
    bool bSuccess = cap.read(frame);
    if (!bSuccess) {
        cout << "Cannot read the frame from video file" << endl;
        break;
    }

    int i = 0;
    recogniseStickersByThreshold(frame, stickers);
    for (auto st : stickers)
    {
        Box box;

        cv::Point sticker1;
        sticker1.x = (*min_element(st.begin(), st.end(), cmpPointX)).x;
        sticker1.y = (*min_element(st.begin(), st.end(), cmpPointY)).y;
        cv::Point sticker2;
        sticker2.x = (*max_element(st.begin(), st.end(), cmpPointX)).x;
        sticker2.y = (*max_element(st.begin(), st.end(), cmpPointY)).y;
        cv::rectangle(frame, Rect(sticker1, sticker2),cv::Scalar(0,250,0),2);

        box.x1 = sticker1.x;
        box.y1 = sticker1.y;
        box.x2 = sticker2.x;
        box.y2 = sticker2.y;
        box.name = std::to_string(i++);

        json j { box };
        std::stringstream ss;
        std::string str;
        ss << j.dump();
        ss >> str;
        mosquitto_publish(mosq, NULL, "test/tl", str.size(), str.c_str(), 0, false);
        //std::cout << j.dump(4) << "\n";
    }
}

```

```

    imshow("MyVideo", frame);
    if(waitKey(30) == 27) {
        break;
    }
}
mosquitto_disconnect(mosq);
mosquitto_destroy(mosq);

mosquitto_lib_cleanup();
return 0;
}
FROM ubuntu:18.04
RUN apt update
RUN apt install python3-opencv -y && apt-get install -y wget
RUN apt install -y build-essential cmake git pkg-config libgtk-3-dev \
    libavcodec-dev libavformat-dev libswscale-dev libv4l-dev \
    libxvidcore-dev libx264-dev libjpeg-dev libpng-dev libtiff-dev \
    gfortran openexr libatlas-base-dev python3-dev python3-numpy \
    libtbb2 libtbb-dev libdc1394-22-dev
RUN mkdir ~/opencv_build && cd ~/opencv_build && git clone
https://github.com/opencv/opencv.git && git clone
https://github.com/opencv/opencv_contrib.git
RUN cd ~/opencv_build/opencv && mkdir build && cd build \
&& cmake -DCMAKE_CXX_FLAGS=-std=c++11 \
    -DCMAKE_BUILD_TYPE=RELEASE \
    -DCMAKE_INSTALL_PREFIX=/usr/local \
    -DBUILD_EXAMPLES=OFF \
    -DBUILD_DOCS=OFF \
    -DBUILD_PERF_TESTS=OFF \
    -DBUILD_TESTS=OFF \
    -DINSTALL_C_EXAMPLES=OFF \
    -DENABLE_PRECOMPILED_HEADERS=OFF \
    -DWITH_OPENMP=ON \
    -DWITH_V4L=ON \
    -DWITH_TBB=ON \
    -DWITH_OPENGL=ON \
    -DWITH_JPEG=ON \
    -DWITH_FFMPEG=ON \
    -DWITH_GSTREAMER=ON \
    -DWITH_OPENCL=ON \
    -DWITH_GPHOTO2=ON \
    -DWITH_LIBV4L=ON \
    -DINSTALL_PYTHON_EXAMPLES=ON \
    -DBUILD_SHARED_LIBS=ON \

```

-DENABLE_CXX11=ON ..

RUN `cd ~/opencv_build/opencv/build && make -j8 && make install`

RUN `apt install nlohmann-json-dev`

RUN `git clone https://github.com/nlohmann/json.git && cd json && cmake -DBuildTests=OFF .`

RUN `apt-get install -y apt-utils && apt-get install software-properties-common -y`

RUN `apt install mosquitto mosquitto-clients -y && apt-get install libmosquitto-dev -y`

RUN `apt-add-repository ppa:mosquitto-dev/mosquitto-ppa && apt-get update`

RUN `apt-get install libssl-dev && apt-mark hold mosquitto`

RUN `apt-get install mosquitto -y --allow-change-held-packages`

RUN `cd / && mkdir project`

RUN `apt install -y tmux vim`

COPY `doc /project`

RUN `cd project && cmake CMakeLists.txt && make`

CMD `bash`

`docker image build -t task6 . $ docker run --net=host --env="DISPLAY" --volume="$HOME/.Xauthority:/root/.Xauthority:rw" task6`

`tmux`

`mosquitto -v -p 1880 $ mosquitto_sub -t test/tl -p 1880`

`cd ./project $ cmake CMakeLists.txt $ make $./make`

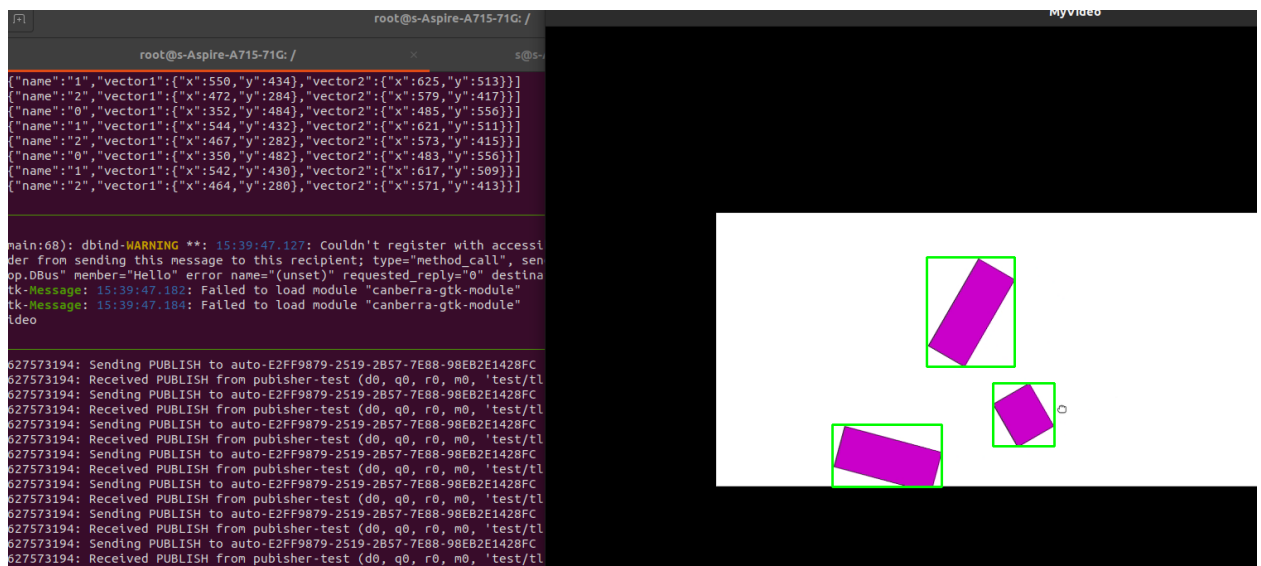


Рисунок 8 — работа mqtt

7. Изучение kafka

Kafka – это платформа, обеспечивающая для хранения огромных объемов данных и взаимодействия с ними. Задача собрать докер с kafka и проверить работоспособность. Результат представлен на рисунке 9.

FROM ubuntu:18.04

RUN apt update

RUN apt install python3-opencv -y && apt-get install -y wget

RUN apt install -y build-essential cmake git pkg-config libgtk-3-dev \
libavcodec-dev libavformat-dev libswscale-dev libv4l-dev \
libxvidcore-dev libx264-dev libjpeg-dev libpng-dev libtiff-dev \
gfortran openexr libatlas-base-dev python3-dev python3-numpy \
libtbb2 libtbb-dev libdc1394-22-dev

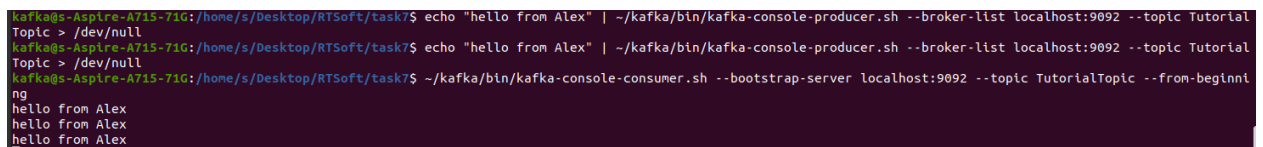
RUN apt-get install -y default-jre

RUN apt install -y curl

RUN echo "1\n 1\n \n\n\n\n y\n" | adduser kafka \&& adduser kafka sudo \&& su -l kafka\&& mkdir ~/Downloads\&& curl "https://downloads.apache.org/kafka/2.6.2/kafka_2.13-2.6.2.tgz" -o ~/Downloads/kafka.tgz\&& mkdir ~/kafka && cd ~/kafka\&& tar -xvzf ~/Downloads/kafka.tgz --strip 1\&& echo "delete.topic.enable = true \n log.dirs=/home/kafka/logs" >> ~/kafka/config/server.properties

RUN echo "[Unit] \n Requires=network.target remote-fs.target \n After=network.target remote-fs.target\ \n [Service] \n Type=simple \n User=kafka \n ExecStart=/home/kafka/kafka/bin/zookeeper-server-start.sh /home/kafka/kafka/config/zookeeper.properties \n ExecStop=/home/kafka/kafka/bin/zookeeper-server-stop.sh \n Restart=on-abnormal \n [Install] \n WantedBy=multi-user.target" >> /etc/systemd/system/zookeeper.service\&& echo "[Unit] \n Requires=zookeeper.service \n After=zookeeper.service \n [Service] \n Type=simple \n User=kafka \n ExecStart=/bin/sh -c 'home/kafka/kafka/bin/kafka-server-start.sh /home/kafka/kafka/config/server.properties > /home/kafka/kafka/kafka.log 2>&1' \n ExecStop=/home/kafka/kafka/bin/kafka-server-stop.sh \n Restart=on-abnormal \n \n [Install] \n WantedBy=multi-user.target" >> /etc/systemd/system/kafka.service

CMD bash



```
kafka@Aspire-A715-71G:/home/s/Desktop/RTSoft/task7$ echo "hello from Alex" | ~/kafka/bin/kafka-console-producer.sh --broker-list localhost:9092 --topic TutorialTopic > /dev/null
kafka@Aspire-A715-71G:/home/s/Desktop/RTSoft/task7$ echo "hello from Alex" | ~/kafka/bin/kafka-console-producer.sh --broker-list localhost:9092 --topic TutorialTopic > /dev/null
kafka@Aspire-A715-71G:/home/s/Desktop/RTSoft/task7$ ~/kafka/bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic TutorialTopic --from-beginning
hello from Alex
hello from Alex
hello from Alex
```

Рисунок 8 — работа kafka

8. Индивидуальный проект – Анализ дорожной разметки

Задача: разработать алгоритм, сигнализирующий куда нужно сместиться транспорту, чтобы остаться в своей полосе.

Для решения поставленной задачи будем использоваться следующие алгоритмы. Программа разработана на языке python3.

OpenCV – (Open Computer Vision) — библиотека компьютерного зрения с открытым исходным кодом, предоставляющая набор типов данных и численных алгоритмов для обработки изображений алгоритмами компьютерного зрения.

Преобразование Хаффа - преобразование Хафа является линейным преобразованием для обнаружения прямых. Прямая может быть задана уравнением $y = mx + b$.

Фильтр Канни – Оператор обнаружения границ изображения. Использует многоступенчатый алгоритм для обнаружения широкого спектра границ в изображениях.

На рисунке 9 изображен входной видео поток, на рисунке 10 — работы фильтра Канни с гауссовым преобразованием. Чтобы наш алгоритм не обрабатывал шумы мы ограничиваем зону видимости объектов, как показано на рисунке 11. Затем с помощью преобразования Хаффа находим полосы дорожной разметки и белой стрелкой указываем куда водитель предпочтительно сместиться. Результат представлен на рисунке 12.

Код программы:

```
import cv2
```

```
import numpy as np
```

```
#Гаусово размытие
```

```
def canny(img):
```

```
    img = cv2.cvtColor(img, cv2.COLOR_BGR2BGRA)
```

```
    blur = cv2.GaussianBlur(img, (5,5), 0)
```

```
    return cv2.Canny(blur, 50, 120)
```

```
def make_coordinates(image, line_parameters): #функция, задающая  
координаты прямой
```

```
    slope, intercept = line_parameters
```

```
    y1 = image.shape[0]
```

```
    y2 = int(y1 * (3/5) + 50)
```

```
    x1 = int((y1 - intercept) / slope)
```

```
    x2 = int((y2 - intercept) / slope)
```

```
    return np.array([x1, y1, x2, y2])
```

```

def average_slope_intercept(image, lines):
    left_fit = []
    right_fit = []
    while lines is not None:
        for line in lines:
            x1, y1, x2, y2 = line.reshape(4)
            parameters = np.polyfit((x1, x2), (y1, y2), 1)
            slope = parameters[0]
            intercept = parameters[1]
            if slope < 0:
                left_fit.append((slope, intercept))
            else:
                right_fit.append((slope, intercept))
        left_fit_average = np.average(left_fit, axis=0)
        left_line = make_coordinates(image, left_fit_average)
        right_fit_average = np.average(right_fit, axis=0)
        right_line = make_coordinates(image, right_fit_average)
        return np.array([left_line, right_line])

def display_lines(image, lines): #функция для отрисовки линий
    line_image = np.zeros_like(image)
    x_l = lines[0][0]
    x_r = lines[1][0]
    dx = 1920 - x_r - x_l
    if lines is not None:
        for x1, y1, x2, y2 in lines:
            cv2.line(line_image, (x1, y1), (x2, y2), (255, 0, 250), 5)
            # линия, указывающая направление желательного смещения
            cv2.line(line_image, (int((x_r + x_l) / 2), 1080), (int((960 - dx)) , 900), (255,
255, 255), 5)
        return line_image

def trapezoid(image):
    height = image.shape[0]
    polygons = np.array([(20, height), (1300, height), (900, 700), (300, 700)])
    mask = np.zeros_like(image)
    cv2.fillPoly(mask, np.array([polygons], dtype=np.int64), 1024)
    masked_image = cv2.bitwise_and(image, mask)
    return masked_image

def main() :
    name_video = "test2.mp4"
    video = cv2.VideoCapture(name_video)

```

```

cv2.waitKey(1)

state = 3; # 0 - обычная картинка
          # 1 - фильтр чёрнобелый
          # 2 - обрезка трапецией
          # 3 - отрисовка линий
          #

while video.isOpened():
    _, frame = video.read()

    cv2.namedWindow(name_video, cv2.WINDOW_NORMAL)
    cv2.resizeWindow(name_video, 640, 360)

    copy_img = np.copy(frame)

    try:
        if state > 0:
            frame = canny(frame)
        if state > 1:
            frame = trapezoid(frame)
        if state > 2:
            lines = cv2.HoughLinesP(frame, 2, np.pi/180, 100, np.array([[]]),
minLineLength = 100, maxLineGap = 50)
            averaged_lines = average_slope_intercept(frame, lines)
            line_image = display_lines(copy_img, averaged_lines)
            combo = cv2.addWeighted(copy_img, 0.8, line_image, 0.5, 1)
            cv2.imshow(name_video, combo)
        if state < 3:
            cv2.imshow(name_video, frame)
    except:
        pass

    wait = cv2.waitKey(1)
    for i in range(0, 4):
        if (wait & 0xFF == (ord('0') + i)):
            state = i
            break

    if (wait & 0xFF == 32):
        while 1:
            wait = cv2.waitKey(1)
            if (wait & 0xFF == 32):
                break

```

```
if (wait & 0xFF == 27):  
    video.release()  
    cv2.destroyAllWindows()  
    break  
main()
```

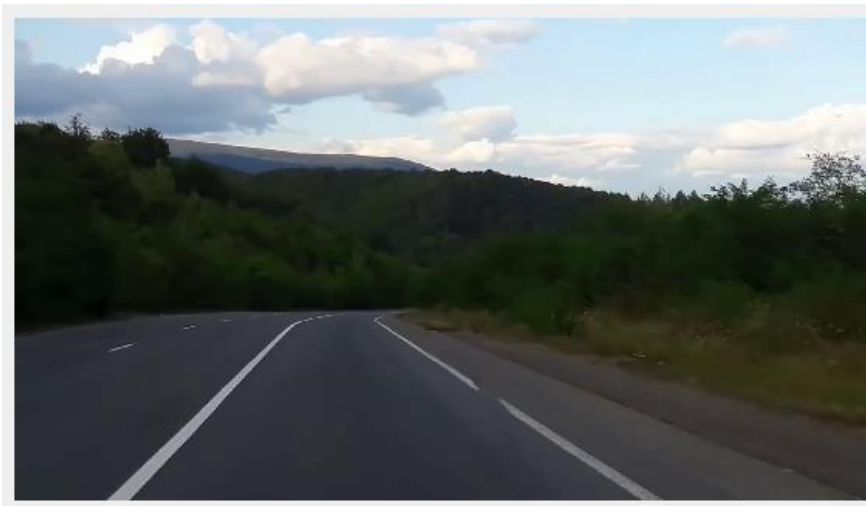


Рисунок 9 — Входной видеопоток

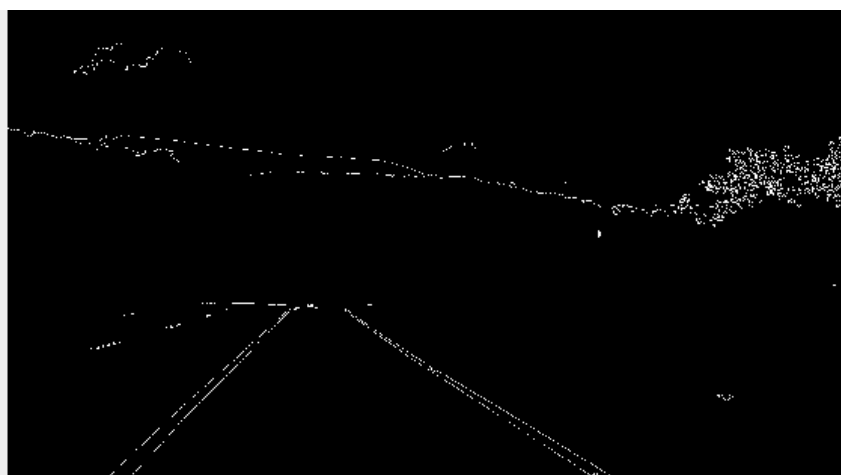


Рисунок 10 — Обработка фильтром Канни

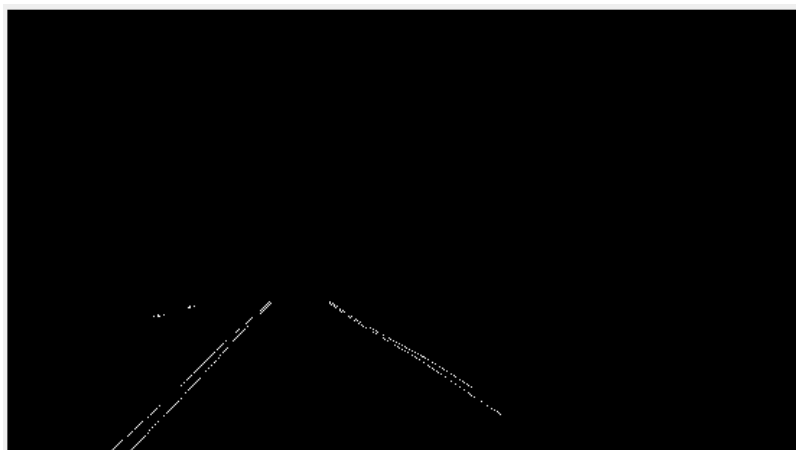


Рисунок 11 — обрезка изображения

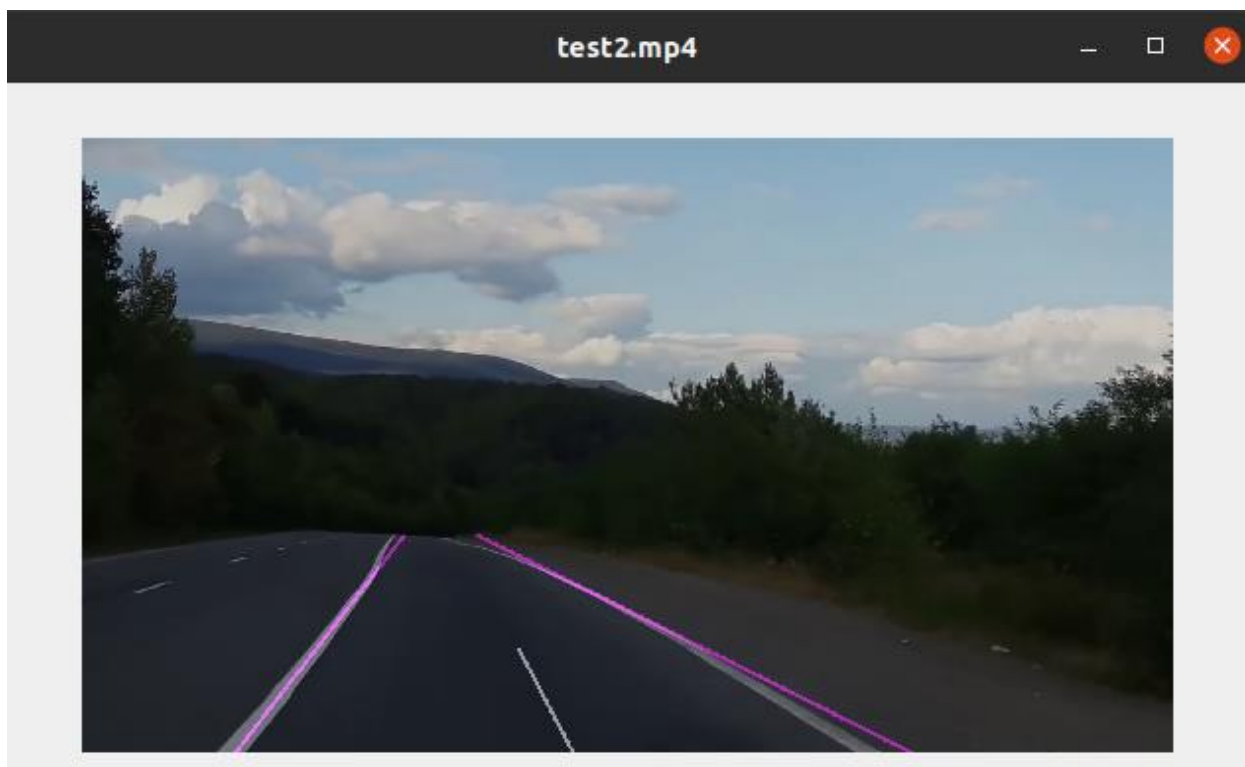


Рисунок 12 — Выходной видеопеток

Заключение

В ходе летней практики были изучены современные технологии и алгоритмы, приобретены навыки работы в ОС Linux, написания драйвера, создание докеров, создания алгоритма компьютерного зрения с использованием библиотеки OpenCV.

Список использованной литературы

1. Лекционные материалы
2. Бьерн Страуструп. Язык программирования C++.
3. <http://docs.opencv.org/3.0-beta/index.html>
4. <https://habr.com/ru/post/106702/>