# 深 圳 大 学 实 验 报 告

课程名称：智能无人系统与边缘计算

实验项目名称：视频流目标检测创新应用

学院：电子与信息工程学院

专业：电子信息工程（文华班）

指导教师：蒙山

报告人：余韦藩 学号：**2020285102** 班级：文华班

实验时间：**2022.12.21-2023.2.18**

实验报告提交时间：**2023** 年 **2** 月 **18** 日

教务部制

实验目的与要求：

**Basic requirements:**

a. Innovate image target detection application, and use YOLO v5 to complete image target training and detection.

b. Get video and image data through RTSP pull stream (you can build RTSP server by yourself), and display video stream content in the window.

**c.** Complete the self-defined image target detection reasoning frame by frame, display the target detection results in the window, and mark different detection targets through different color boxes.

**Additional requirements:**

a. Use stream processing framework, such as GStreamer, and YOLO v5 to build a complete image target detection application.

b. Get video stream data directly from USB camera or network IP camera to complete innovative application of image target detection.

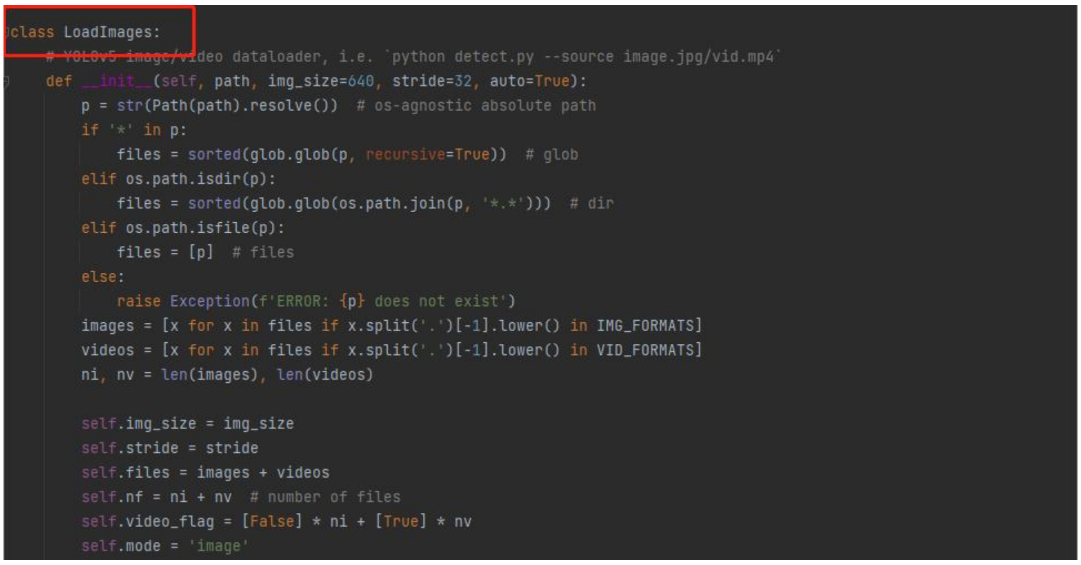
c. Use the common NVIDIA Jetson Nano, or raspberry pie, or Horizon X3 pie, or Ruixin micro-series AI computing board to develop and realize the corresponding image target detection function.

**Experiment process and experiment result:**

**Basic requirements:**

a. **Innovate image target detection application, and use YOLO v5 to complete image target training and detection.**

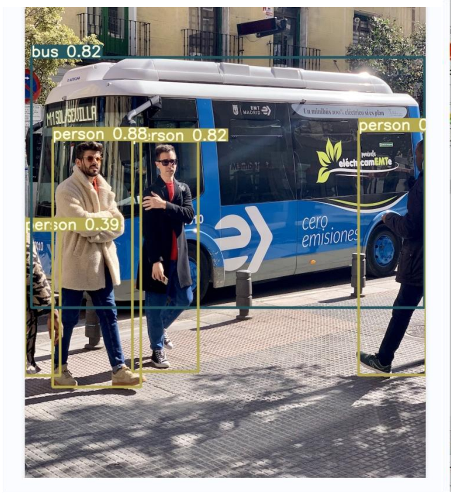
**Experiment process:** Here we write a LoadImages class object to deal with the image input or video input as shown in the following figure. We have write some notes in the program so we don't explain it in detail. The main program part of the LoadImages is judge whether the input is image or video. Then save all the frames in the class object, resize them and convert to the format of RGB. Combined with the yolo-v5 program, we achieve perform object detection for image or video.

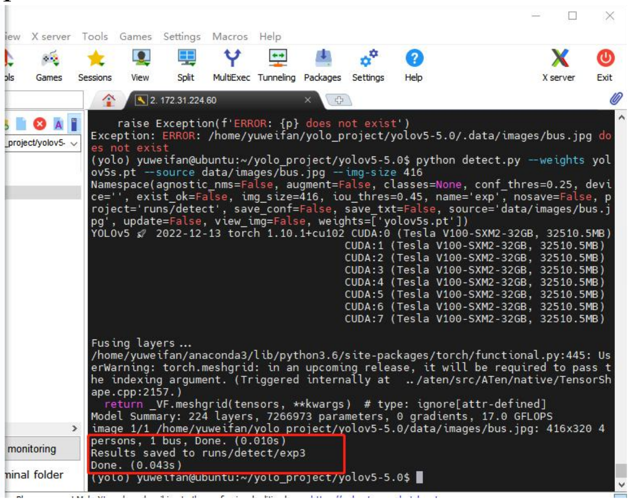


```
class LoadImages:
    # YOLOv5 image/video dataloader, i.e. `python detect.py --source image.jpg/vid.mp4`
    def __init__(self, path, img_size=640, stride=32, auto=True):
        p = str(Path(path).resolve())  # os-agnostic absolute path
        if '*' in p:
            files = sorted(glob.glob(p, recursive=True))  # glob
        elif os.path.isdir(p):
            files = sorted(glob.glob(os.path.join(p, '*.*')))  # dir
        elif os.path.isfile(p):
            files = [p]  # files
        else:
            raise Exception(f'ERROR: {p} does not exist')
        images = [x for x in files if x.split('.')[-1].lower() in IMG_FORMATS]
        videos = [x for x in files if x.split('.')[-1].lower() in VID_FORMATS]
        ni, nv = len(images), len(videos)

        self.img_size = img_size
        self.stride = stride
        self.files = images + videos
        self.nf = ni + nv  # number of files
        self.video_flag = [False] * ni + [True] * nv
        self.mode = 'image'
```

**Experiment result:**

The object detection for example image is shown in the following figure. Here we observe the inter time is tiny. What's more, I achieve object detection for mp4 video. The original mp4 video(MP4 原视频) and the process mp4 video(MP4 视频检测) have been attached to this report.

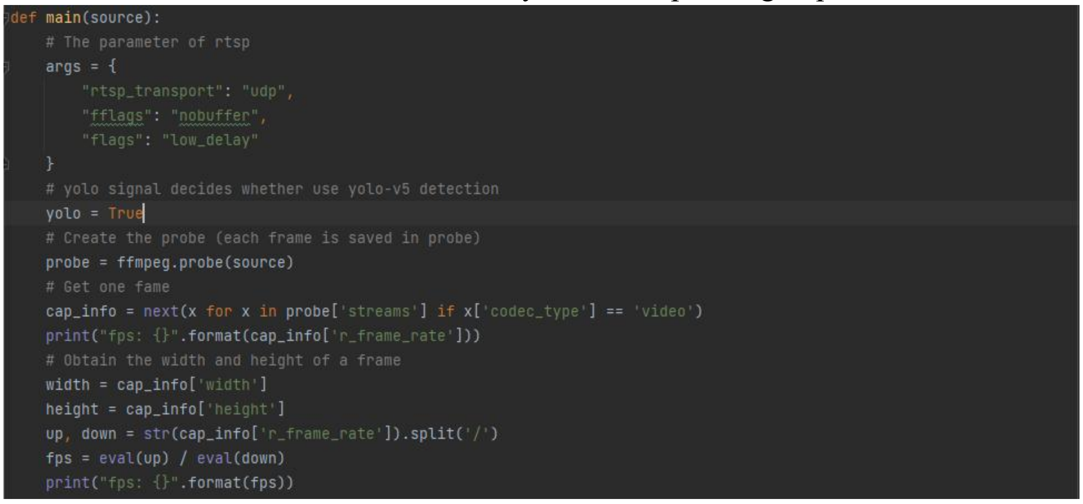




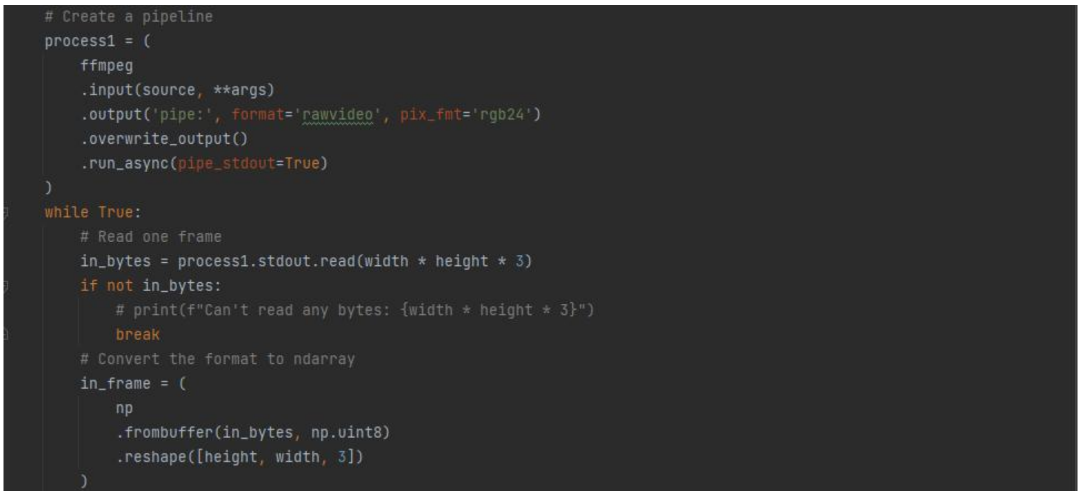
```
    raise Exception(f'ERROR: {p} does not exist')
Exception: ERROR: /home/yuweifan/yolo_project/yolov5-5.0/.data/images/bus.jpg do
es not exist
(yolo) yuweifan@ubuntu:~/yolo_project/yolov5-5.0$ python detect.py --weights yol
ov5s.pt --source data/images/bus.jpg --img-size 416
Namespace(agnostic_nms=False, augment=False, classes=None, conf_thres=0.25, devi
ce='', exist_ok=False, img_size=416, iou_thres=0.45, name='exp', nosave=False, p
roject='runs/detect', save_conf=False, save_txt=False, source='data/images/bus.j
pg', update=False, view_img=False, weights=['yolov5s.pt'])
YOLOv5 🚀 2022-12-13 torch 1.10.1+cu102 CUDA:0 (Tesla V100-SXM2-32GB, 32510.5MB)
                                       CUDA:1 (Tesla V100-SXM2-32GB, 32510.5MB)
                                       CUDA:2 (Tesla V100-SXM2-32GB, 32510.5MB)
                                       CUDA:3 (Tesla V100-SXM2-32GB, 32510.5MB)
                                       CUDA:4 (Tesla V100-SXM2-32GB, 32510.5MB)
                                       CUDA:5 (Tesla V100-SXM2-32GB, 32510.5MB)
                                       CUDA:6 (Tesla V100-SXM2-32GB, 32510.5MB)
                                       CUDA:7 (Tesla V100-SXM2-32GB, 32510.5MB)

Fusing layers...
/home/yuweifan/anaconda3/lib/python3.6/site-packages/torch/functional.py:445: Us
erWarning: torch.meshgrid: in an upcoming release, it will be required to pass t
he indexing argument. (Triggered internally at  ../aten/src/ATen/native/TensorSh
ape.cpp:2157.)
  return _VF.meshgrid(tensors, **kwargs)  # type: ignore[attr-defined]
Model Summary: 224 layers, 7266973 parameters, 0 gradients, 17.0 GFLOPS
image 1/1 /home/yuweifan/yolo_project/yolov5-5.0/data/images/bus.jpg: 416x320 4
persons, 1 bus, Done. (0.010s)
Results saved to runs/detect/exp3
Done. (0.043s)
(yolo) yuweifan@ubuntu:~/yolo_project/yolov5-5.0$
```

b. **Get video and image data through RTSP pull stream (you can build RTSP server by yourself), and display video stream content in the window.**

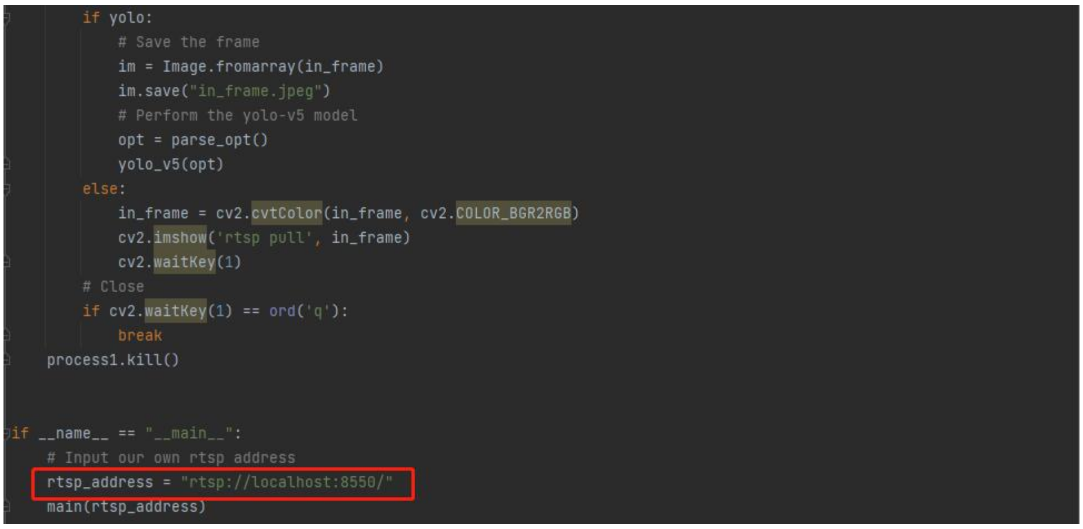
**Experiment process:** First, we should program to achieve rtsp pull stream. The core is to extract each frame in the video stream and display it in the screen. The program is shown in the following. Some notes are marked in it so we will not explain it in detail. It is worth to say that we use a yolo signal variance to control the output from original video stream to processed(object detection) video stream. What's more, in order to display normal for the cv2 format image, we should convert it to the format of RGB before display it. The output of the video stream is equivalent to display each frame in the video stream. In order to play the video stream that we created, we should modify the corresponding rtsp address.



```python
def main(source):
    # The parameter of rtsp
    args = {
        "rtsp_transport": "udp",
        "fflags": "nobuffer",
        "flags": "low_delay"
    }
    # yolo signal decides whether use yolo-v5 detection
    yolo = True
    # Create the probe (each frame is saved in probe)
    probe = ffmpeg.probe(source)
    # Get one fame
    cap_info = next(x for x in probe['streams'] if x['codec_type'] == 'video')
    print("fps: {}".format(cap_info['r_frame_rate']))
    # Obtain the width and height of a frame
    width = cap_info['width']
    height = cap_info['height']
    up, down = str(cap_info['r_frame_rate']).split('/')
    fps = eval(up) / eval(down)
    print("fps: {}".format(fps))
```

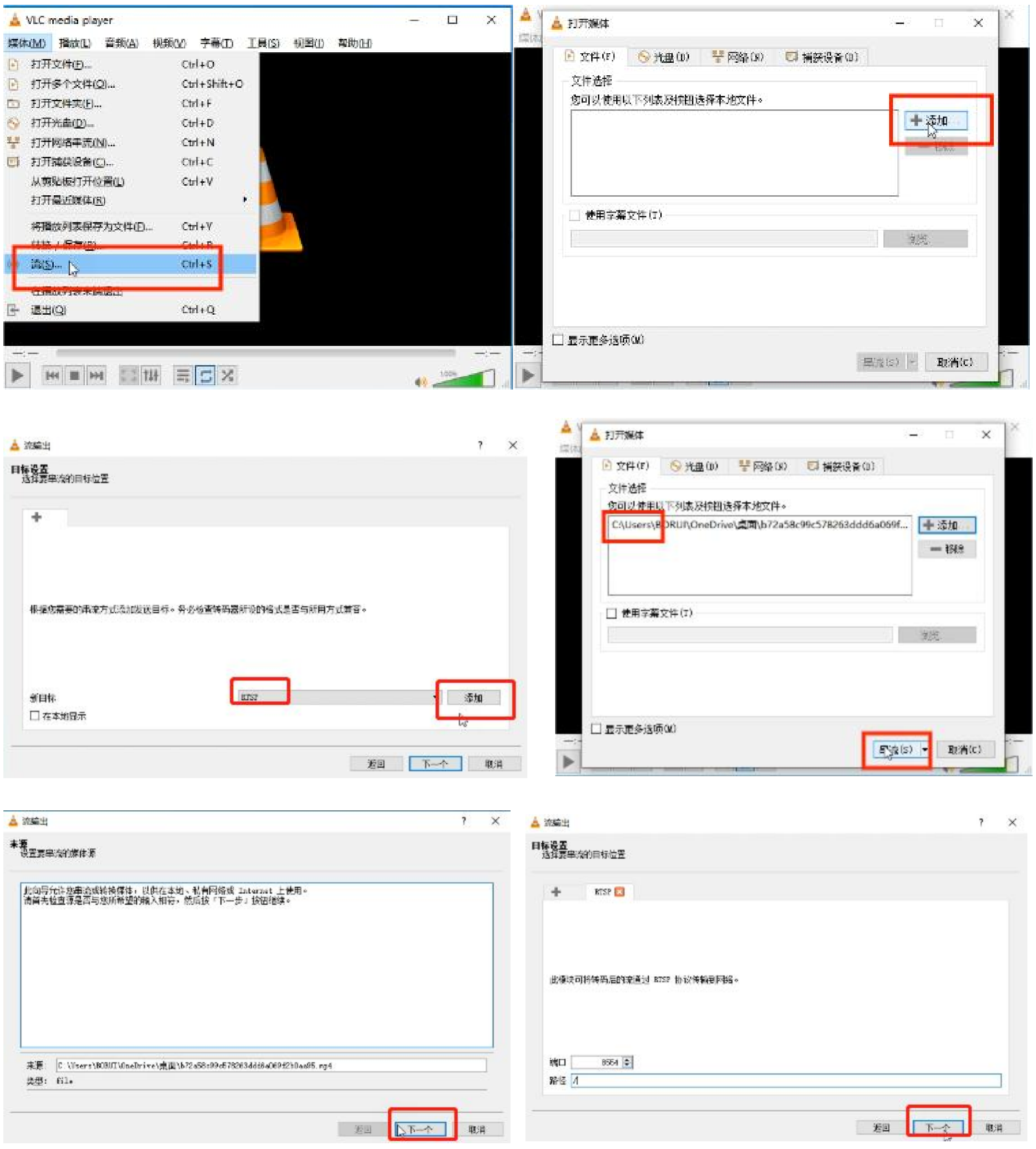


```python
    # Create a pipeline
    process1 = (
        ffmpeg
        .input(source, **args)
        .output('pipe:', format='rawvideo', pix_fmt='rgb24')
        .overwrite_output()
        .run_async(pipe_stdout=True)
    )
    while True:
        # Read one frame
        in_bytes = process1.stdout.read(width * height * 3)
        if not in_bytes:
            # print(f"Can't read any bytes: {width * height * 3}")
            break
        # Convert the format to ndarray
        in_frame = (
            np
            .frombuffer(in_bytes, np.uint8)
            .reshape([height, width, 3])
        )
```



```
        if yolo:
            # Save the frame
            im = Image.fromarray(in_frame)
            im.save("in_frame.jpeg")
            # Perform the yolo-v5 model
            opt = parse_opt()
            yolo_v5(opt)
        else:
            in_frame = cv2.cvtColor(in_frame, cv2.COLOR_BGR2RGB)
            cv2.imshow('rtsp pull', in_frame)
            cv2.waitKey(1)
        # Close
        if cv2.waitKey(1) == ord('q'):
            break
    process1.kill()


if __name__ == "__main__":
    # Input our own rtsp address
    rtsp_address = "rtsp://localhost:8550/"
    main(rtsp_address)
```

In addition, we should build a rtsp server. Here we use the VLC software to generate our own rtsp server. All the steps are shown in the following figure.



您可以使用以下列表及按钮选择本地文件。

C:\Users\BORUI\OneDrive\桌面\b72a58c99c578263ddd6a069f...

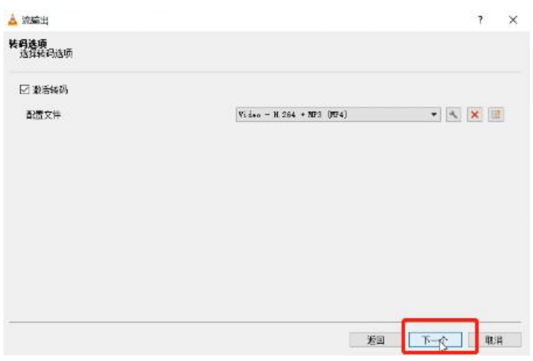
根据您需要的串流方式添加发送目标。务必检查转码器所设的格式是否与所用方式兼容。

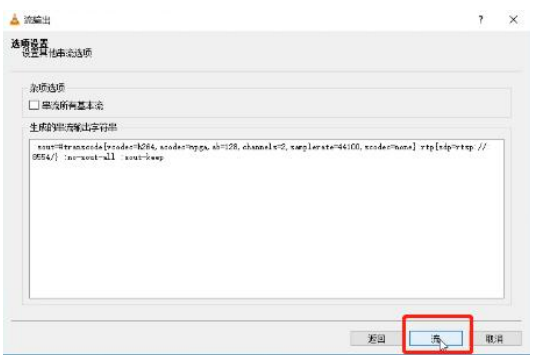
此向导允许您串流或转换媒体，以供在本地、私有网络或 Internet 上使用。
请首先检查源是否与您所希望的输入相符，然后按「下一步」按钮继续。

来源: C:\Users\BORUI\OneDrive\桌面\b72a58c99c578263ddd6a069f2b0aa85.mp4
类型: file

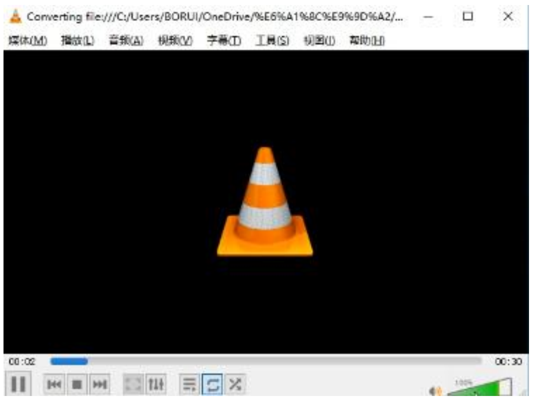
此模块可将转码后的流通过 RTSP 协议传输到网络。

端口 8554
路径 /



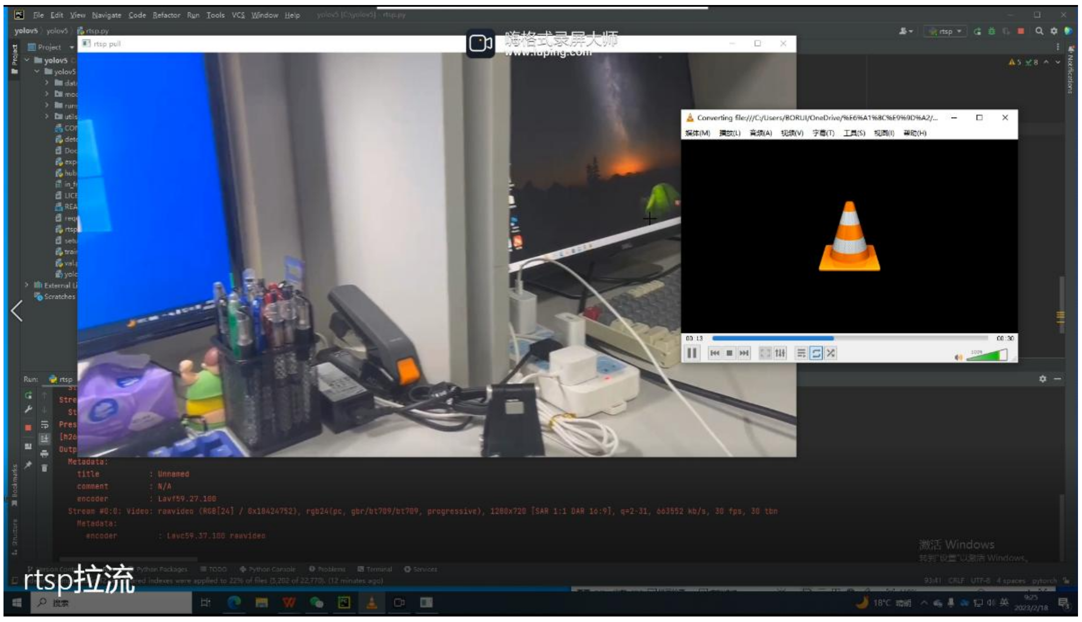


```
:sout=#transcode{vcodec=h264,acodec=mpga,ab=128,channels=2,samplerate=44100,scodec=none}:rtp{sdp=rtsp://
:8554/} :no-sout-all :sout-keep
```


Converting file:///C:/Users/BORUI/OneDrive/%E6%A1%8C%E9%9D%A2/...
媒体(M) 播放(L) 音频(A) 视频(V) 字幕(T) 工具(S) 视图(I) 帮助(H)
00:02
00:30

After finish all the two process, we achieve get video and image data through RTSP pull stream.

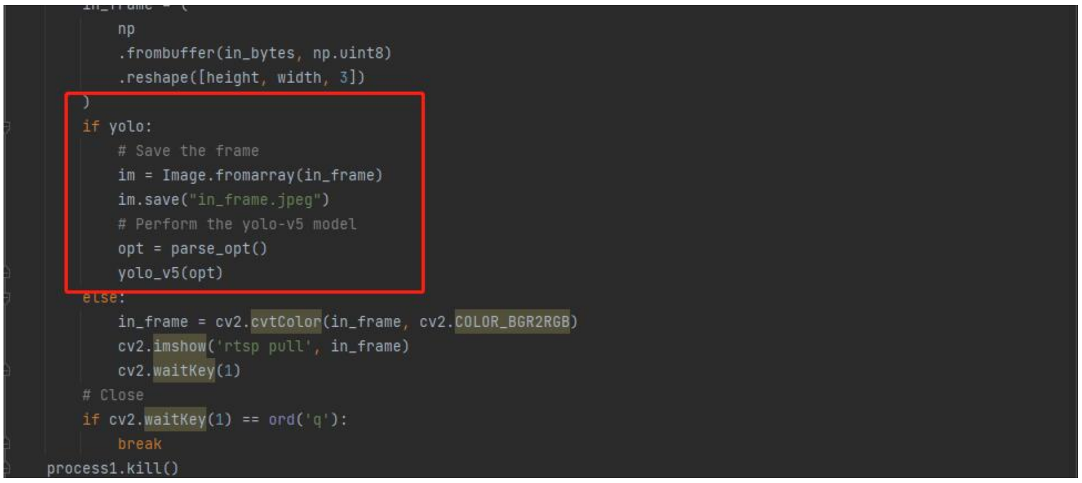
**Experiment result:** The original mp4 video(rtsp 原视频) and the pull video stream(rtsp 拉流) have been attached to this report. Here is a screen shoot for the pull video stream.


rtsp pull
Converting file:///C:/Users/BORUI/OneDrive/%E6%A1%8C%E9%9D%A2/...
Run: rtsp
rtsp拉流
激活 Windows

```
  Metadata:
    title           : Unnamed
    comment         : N/A
    encoder         : Lavf59.27.100
  Stream #0:0: Video: rawvideo (RGB[24] / 0x18424752), rgb24(pc, gbr/bt709/bt709, progressive), 1280x720 [SAR 1:1 DAR 16:9], q=2-31, 663552 kb/s, 30 fps, 30 tbn
    Metadata:
      encoder         : Lavc59.37.100 rawvideo
```

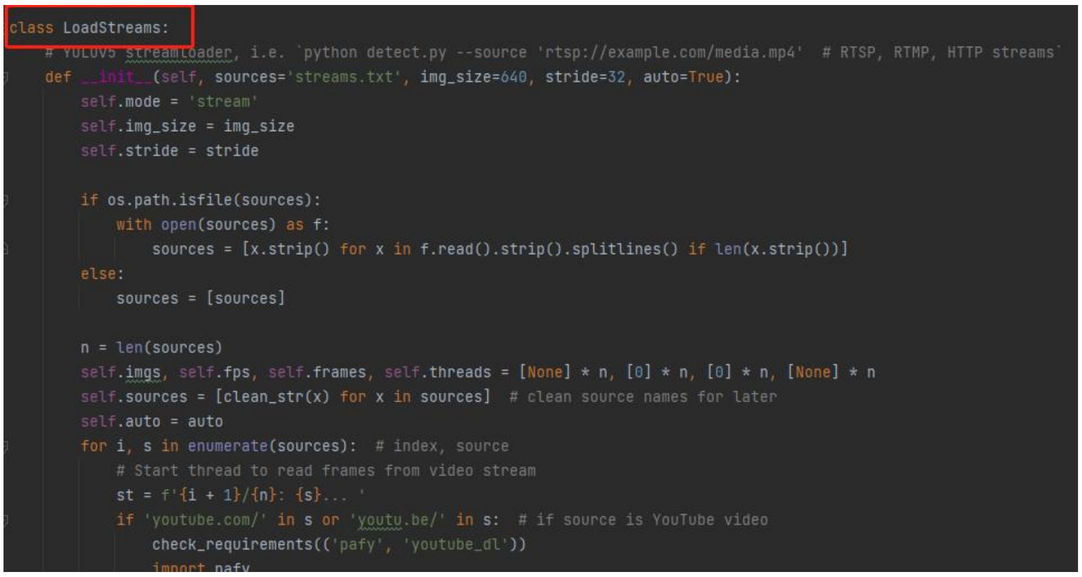
**c. Complete the self-defined image target detection reasoning frame by frame, display the target detection results in the window, and mark different detection targets through different color boxes.**

**Experiment process:** We find out the part of extracting each frame in the video stream program. If we want to perform object detection for the video stream, we do perform object detection for each frame. So before display the original frame, we use yolo-v5 model to perform object detection for each frame and output each processed frame in the screen. The following shows the corresponding program for each frame.



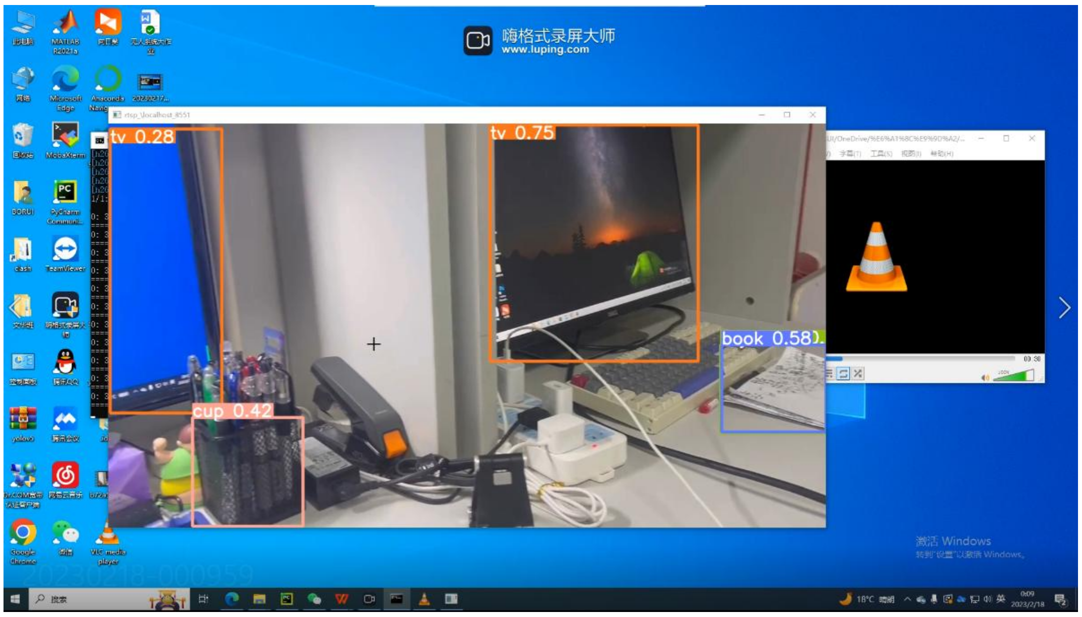
```
    in_frame = (
        np
        .frombuffer(in_bytes, np.uint8)
        .reshape([height, width, 3])
    )
    if yolo:
        # Save the frame
        im = Image.fromarray(in_frame)
        im.save("in_frame.jpeg")
        # Perform the yolo-v5 model
        opt = parse_opt()
        yolo_v5(opt)
    else:
        in_frame = cv2.cvtColor(in_frame, cv2.COLOR_BGR2RGB)
        cv2.imshow('rtsp pull', in_frame)
        cv2.waitKey(1)
    # Close
    if cv2.waitKey(1) == ord('q'):
        break
process1.kill()
```

Here we write a LoadStreams class object to deal with the video stream input as shown in the following figure. We have write some notes in the program so we don't explain it in detail. The main program part of the LoadStreams is read each frame from video stream and save them in the class object.



```
class LoadStreams:
    # YOLOv5 streamloader, i.e. `python detect.py --source 'rtsp://example.com/media.mp4'  # RTSP, RTMP, HTTP streams`
    def __init__(self, sources='streams.txt', img_size=640, stride=32, auto=True):
        self.mode = 'stream'
        self.img_size = img_size
        self.stride = stride

        if os.path.isfile(sources):
            with open(sources) as f:
                sources = [x.strip() for x in f.read().strip().splitlines() if len(x.strip())]
        else:
            sources = [sources]

        n = len(sources)
        self.imgs, self.fps, self.frames, self.threads = [None] * n, [0] * n, [0] * n, [None] * n
        self.sources = [clean_str(x) for x in sources]  # clean source names for later
        self.auto = auto
        for i, s in enumerate(sources):  # index, source
            # Start thread to read frames from video stream
            st = f'{i + 1}/{n}: {s}... '
            if 'youtube.com/' in s or 'youtu.be/' in s:  # if source is YouTube video
                check_requirements(('pafy', 'youtube_dl'))
                import pafy
```

**Experiment result:** The original mp4 video(rtsp 原视频) and the processed pull video stream(rtsp 拉流检测) have been attached to this report. Here is a screen shoot for the processed pull video stream. From the figure, we observe that the frame has been marked by different color of rectangle which indicates our successfully performance. However, the video stream may unstable sometime and appears drop frame phenomenon. I support the reason of this problem may be related to the cv2 package.

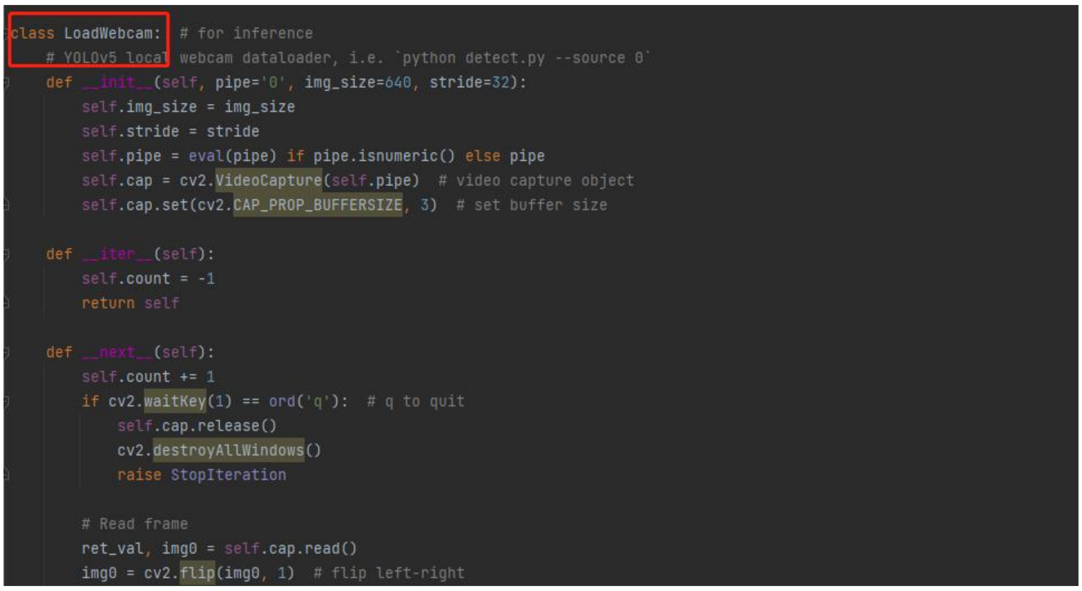


**Additional requirements:**

**a. Get video stream data directly from USB camera or network IP camera to complete innovative application of image target detection.**

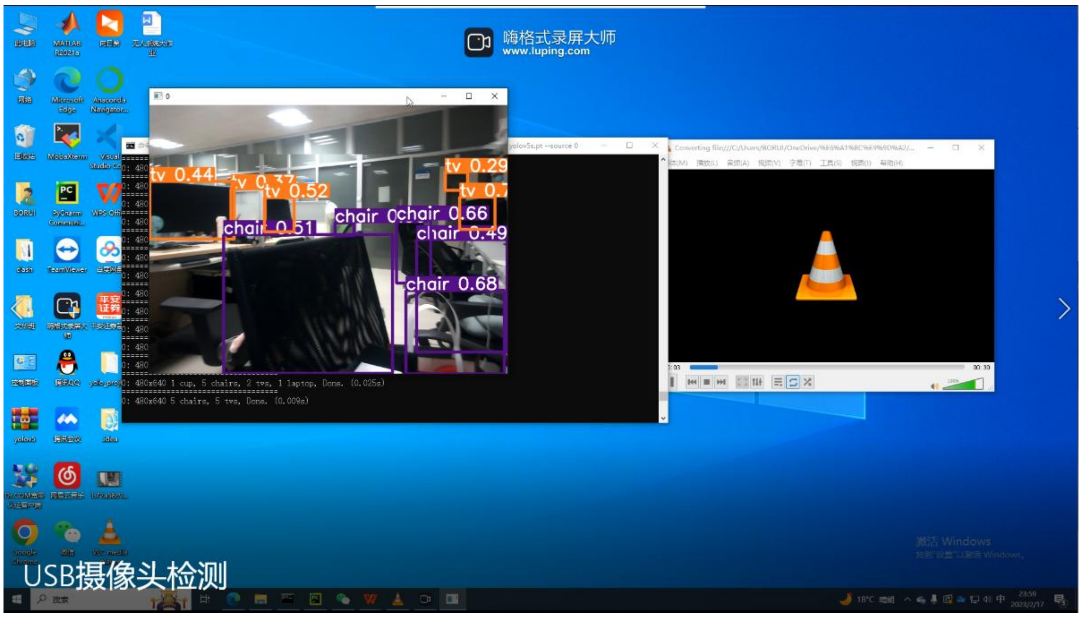
**Experiment process:** I purchase a USB camera from online to achieve this task. The following figure shows the camera.

Here we write a Loadwebcam class object to deal with the external device input as shown in the following figure. We have write some notes in the program so we don't explain it in detail. Combined with the yolo-v5 program, we achieve performing object detection for stream.



```
class LoadWebcam:  # for inference
    # YOLOv5 local webcam dataloader, i.e. `python detect.py --source 0`
    def __init__(self, pipe='0', img_size=640, stride=32):
        self.img_size = img_size
        self.stride = stride
        self.pipe = eval(pipe) if pipe.isnumeric() else pipe
        self.cap = cv2.VideoCapture(self.pipe)  # video capture object
        self.cap.set(cv2.CAP_PROP_BUFFERSIZE, 3)  # set buffer size

    def __iter__(self):
        self.count = -1
        return self

    def __next__(self):
        self.count += 1
        if cv2.waitKey(1) == ord('q'):  # q to quit
            self.cap.release()
            cv2.destroyAllWindows()
            raise StopIteration

        # Read frame
        ret_val, img0 = self.cap.read()
        img0 = cv2.flip(img0, 1)  # flip left-right
```

**Experiment result:** The processed video stream(USB 摄像头检测) have been attached to this report. Here is a screen shoot for the processed pull video stream. With the movement of the camera, the real-time object detection is performed smoothly which proves that our success.

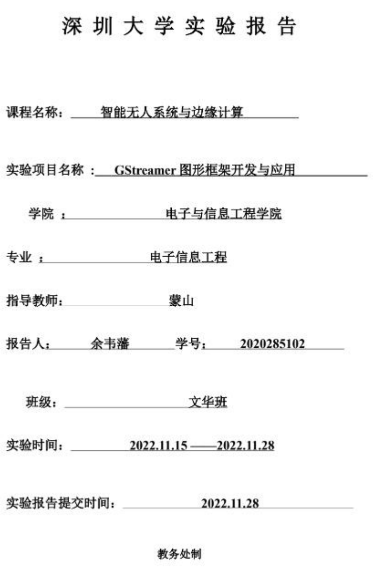


```
0: 480x640 1 cup, 5 chairs, 2 tvs, 1 laptop, Done. (0.025s)
0: 480x640 5 chairs, 5 tvs, Done. (0.009s)
```

**b. Use stream processing framework, such as GStreamer, and YOLO v5 to build a complete image target detection application.**

**Experiment process:** I have been finished this task. However, last week my SSD fail to work and all the data in it including the former finished experiment report and all the result has been lost. If I finish this task again, I should reset the environment of QT and Gstreamer. It is too time consuming that I can't finish it on time. So in the rest of the part, I will briefly introduce the idea to achieve this task.

In the former experiment six, we have the basic of achieving extracting each frame under Gstreamer framework(in this experiment, I successfully achieve all the task) as shown in the following figure.



深 圳 大 学 实 验 报 告

课程名称：智能无人系统与边缘计算

实验项目名称 ：GStreamer 图形框架开发与应用

学院 ：电子与信息工程学院

专业 ：电子信息工程

指导教师：蒙山

报告人：余韦藩　学号：2020285102

班级：文华班

实验时间：2022.11.15 ——2022.11.28

实验报告提交时间：2022.11.28

教务处制

**Aim of Experiment:**

A. Familiar with the QtCreator and Gstreamer framework.
B. Learn how to play a video stream under the Gstreamer framework.
C. Learn some operations for the real time video stream ynder the Gstreamer framework.
D. Learn how to create a GUI window and apply some trigger functions for the bottoms in the GUI window under the QtCreator framework.

**Experiment Content:**

1、安装 QT 和 GStreamer 开发环境，实现对例子程序的编译运行与调试；
2、实现视频码流实时播放；
3、实现实时图像抽帧，并对每一帧图像进行个性化字幕叠加处理；
4、改变图像抽帧速率，在新的窗口实现缓冲队列的连续图像帧 1 秒以上延时显示，与原视频窗口同屏对比。

**Experiment Process:**

**A. Install the QT and GStreamer development environment, then run and debug the example programs.**

First we create the container (select pytorch-gui version) and download all the install package as shown in the following figure in the root directory.

```
cmake-3.24.3-windows-x86_64.msi            2022/11/14 14:43   Windows Install...      29,064 KB
gstreamer-1.0-devel-mingw-x86-1.20...      2022/11/15 9:29    Windows Install...     145,028 KB
gstreamer-1.0-devel-msvc-x86_64-1...       2022/10/6 16:49    Windows Installer 程序包 [illegible]04 KB
gstreamer-1.0-mingw-x86_64-1.20.3...       2022/11/15 9:20    Windows Install...     217,760 KB
gstreamer-1.0-msvc-x86_64-1.20.3.msi       2022/11/15 9:16    Windows Install...     106,876 KB
qt-opensource-linux-x64-5.14.2.run         2022/11/14 15:07   RUN 文件              1,304,402...
qt-opensource-windows-x86-5.14.2.e...      2022/11/14 15:27   应用程序              2,391,399...
```

Then I try to run the Qt and open the example program. At the beginning, the include<gtk/gtk.h> raises error "no such file or directory" in the qtoverlay.h file, so I search from Internet and find the solution as shown in the following figure.

gtk/gtk.h: no such file or directory

```
test/[illegible]_test_gui.c:3:10: fatal error: gtk/gtk.h: No such file or directory
    3 | #include <gtk/gtk.h>
compilation terminated.
```

```
$ sudo apt-get install libgtk2.0-dev
$ sudo apt-get install gnome-devel
```

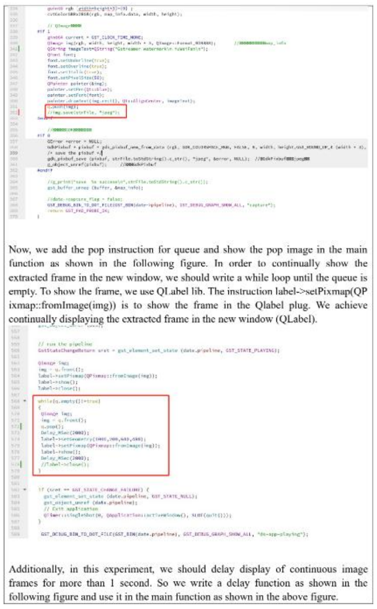
Let us look back to the idea. Each frame of the video stream is pass through pad, so we continuous extract each frame through pad and save them in a queue as shown in the following figure. Then after adding caption, we display them continuous by cv2 window. In order to achieve this task, we only replace caption operation to perform yolo-v5 detection for each frame. Save each processed frame in the queue and display it in original rate.

```
    }
    return TRUE;
}

void PlayerWindow::slotGrabWidgetScreen()
{
    date->capture_flag = true;
    //QMessageBox::information(this, "Grab", "Save jpg succese!",QMessageBox::Ok);
}


//create window2
PlayerWindow2::PlayerWindow2()
{
    label = new QLabel("Label");
    playerLayout = new QVBoxLayout;
    playerLayout->addWidget(label);
}
```

It's easy to guess that the capture_flag control whether we capture the current frame in the pad. And we look back to the extracted image frame processing function as shown in the following figure. We find that once we finish adding caption, the capture_flag is set to false. So in order to continuously extract the frame in the video stream, we could always set the capture_flag to true. In order words, we should not ~~set the capture_flag to false after p~~rocessing the image frame. So I delete this codes to achieve continuous extraction. However, in this experiment, we not only achieve continuous extraction, but also play the extracted image frame in the news window. Consequently, we should use the queue to save the extracted image frame that added caption. In the following part, we will focus on the queue.

```
[illegible]
```

Search from the Internet, I find that in the STL container, there has queue data structure. So in the following part, we will use this queue lib.



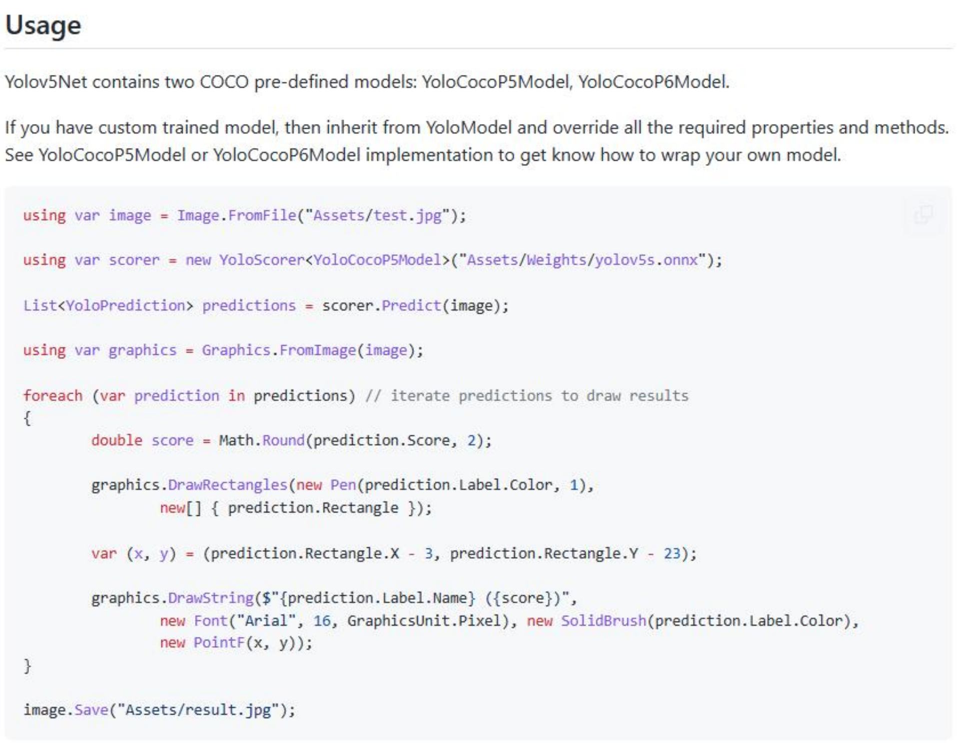
```
[illegible]
```

Now, we add the pop instruction for queue and show the pop image in the main function as shown in the following figure. In order to continually show the extracted frame in the new window, we should write a while loop until the queue is empty. To show the frame, we use QLabel lib. The instruction label->setPixmap(QPixmap::fromImage(img)) is to show the frame in the Qlabel plug. We achieve continually displaying the extracted frame in the new window (QLabel).

```
[illegible]
```

Additionally, in this experiment, we should delay display of continuous image frames for more than 1 second. So we write a delay function as shown in the following figure and use it in the main function as shown in the above figure.

Here we should search yolo-v5 C++ version and perform it according to the following figure.



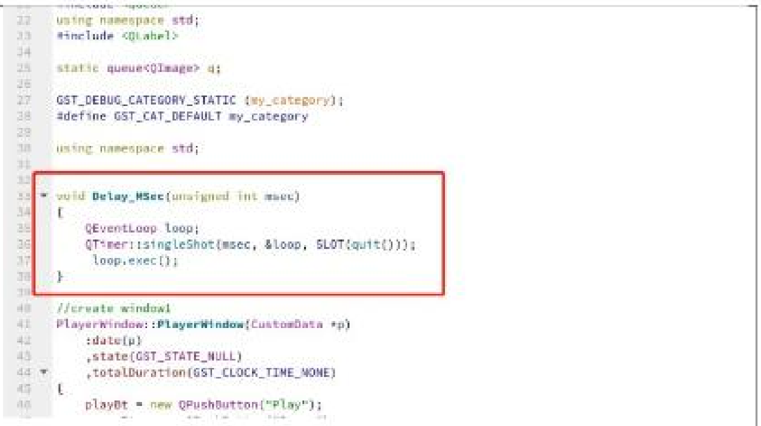
## Usage

Yolov5Net contains two COCO pre-defined models: YoloCocoP5Model, YoloCocoP6Model.

If you have custom trained model, then inherit from YoloModel and override all the required properties and methods. See YoloCocoP5Model or YoloCocoP6Model implementation to get know how to wrap your own model.

```
using var image = Image.FromFile("Assets/test.jpg");

using var scorer = new YoloScorer<YoloCocoP5Model>("Assets/Weights/yolov5s.onnx");

List<YoloPrediction> predictions = scorer.Predict(image);

using var graphics = Graphics.FromImage(image);

foreach (var prediction in predictions) // iterate predictions to draw results
{
        double score = Math.Round(prediction.Score, 2);

        graphics.DrawRectangles(new Pen(prediction.Label.Color, 1),
                new[] { prediction.Rectangle });

        var (x, y) = (prediction.Rectangle.X - 3, prediction.Rectangle.Y - 23);

        graphics.DrawString($"{prediction.Label.Name} ({score})",
                new Font("Arial", 16, GraphicsUnit.Pixel), new SolidBrush(prediction.Label.Color),
                new PointF(x, y));
}

image.Save("Assets/result.jpg");
```

**Experiment result:** The result is similar to the experiment six. It is shame to unable to display the original experiment result.



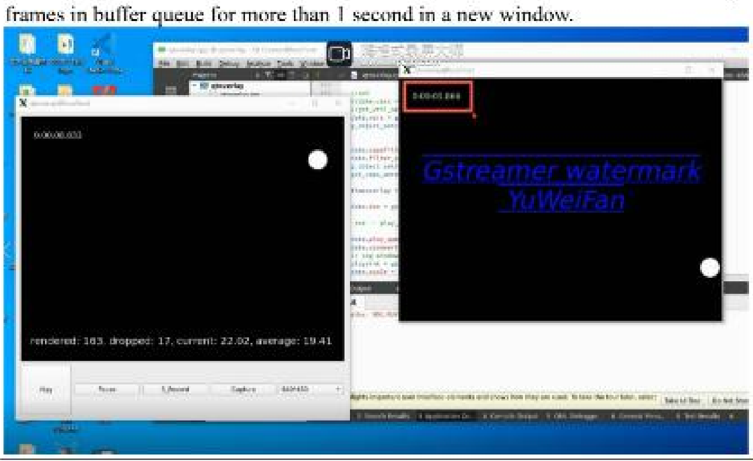
```
using namespace std;
#include <QLabel>

static queue<QImage> q;

GST_DEBUG_CATEGORY_STATIC (my_category);
#define GST_CAT_DEFAULT my_category

using namespace std;


void Delay_MSec(unsigned int msec)
{
    QEventLoop loop;
    QTimer::singleShot(msec, &loop, SLOT(quit()));
     loop.exec();
}

//create window1
PlayerWindow::PlayerWindow(CustomData *p)
    :date(p)
    ,state(GST_STATE_NULL)
    ,totalDuration(GST_CLOCK_TIME_NONE)
{
    playBt = new QPushButton("Play");
```

**Data Logging and Processing:**

In this experiment, there is no need to log the data and perform processing. All the experimental process has been illustrated in the former part.

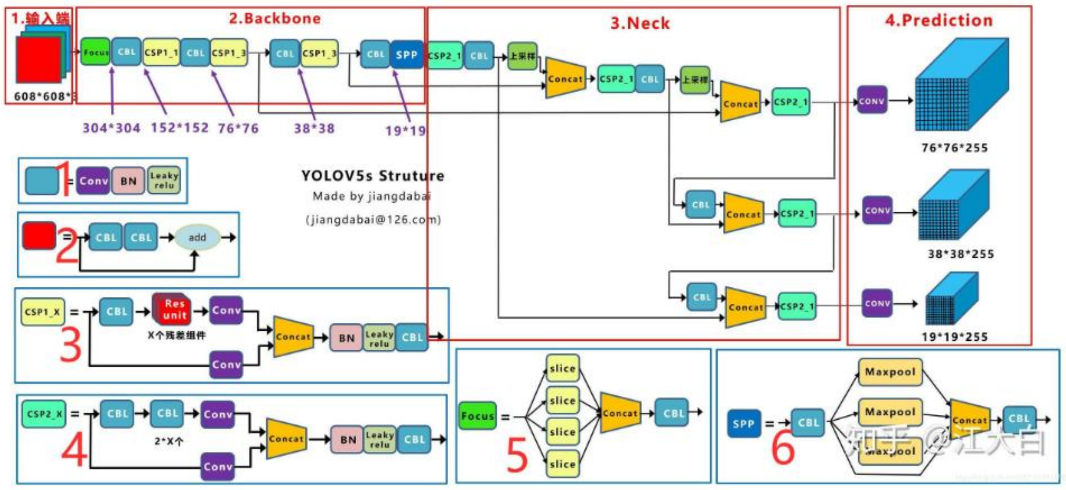
**Experimental Results and Analysis:**

The experiment result video for the forth question has been attached in the BB. In the following figure, the left window shows the original video and the right window shows the continual sequence of extracted image frames. From the video, we observe that time change of right window is slow than the left window and the motion speed of the write ball in the left window is slower than that in the right window, which indicates that we successfully delay display of continuous image frames in buffer queue for more than 1 second in a new window.



**Experiment conclusion:**

Through this experiment, I deeply understand Yolov5 neural network from the following perspectives: network structure, run and application as shown in the following figure.



First, we use the yolo-v5 to perform object detection for a given image. In addition, I understand the rtsp protocol which is often used to pull the video stream. Since the video consists of sequential images, the rest task require us to perform object detection for video stream. In this experiment, we extract each frame in the rtsp stream and perform object detection by yolo-v5. Moreover, we collect the external USB capture video stream and also perform object detection for each frame. The object recorded video is saved in convince to observe. Finally, we use the gstreamer framework to transmit the video stream and perform object detection for each frame. All the task is well finish and the corresponding result has been attached to the report.

**指导教师批阅意见：**

**成绩评定：**

| |
|---|
| **指导教师签字：**<br>年　　月　　日 |
| **备注：** |

注：1、报告内的项目或内容设置，可根据实际情况加以调整和补充。

2、教师批改学生实验报告时间应在学生提交实验报告时间后 10 日内。