Instance segmentation

This example demonstrates instance segmentation. It uses the yolov5n_seg model.

1. Detect the video

Input in the terminal,

```
cd hailo-rpi5-examples/
```

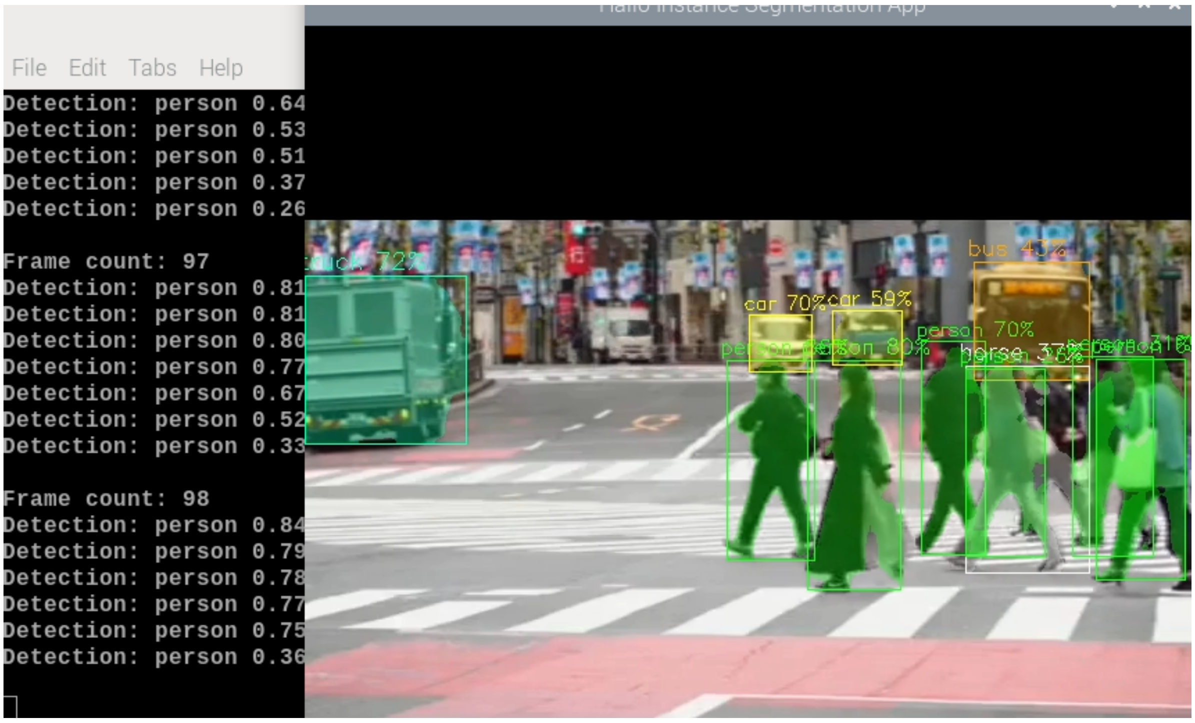
Then enter the command to enter the virtual environment

```
source setup_env.sh
```

Enter the command. Run the video to detect

```
python basic_pipelines/instance_segmentation.py --input resources/detection0.mp4
```

Where the .mp4 suffix indicates the path of the video



```
Detection: person 0.64
Detection: person 0.53
Detection: person 0.51
Detection: person 0.37
Detection: person 0.26

Frame count: 97
Detection: person 0.81
Detection: person 0.81
Detection: person 0.80
Detection: person 0.77
Detection: person 0.67
Detection: person 0.52
Detection: person 0.33

Frame count: 98
Detection: person 0.84
Detection: person 0.79
Detection: person 0.78
Detection: person 0.77
Detection: person 0.75
Detection: person 0.36
```

To close it, press ctrl+c.

Contents in this example:

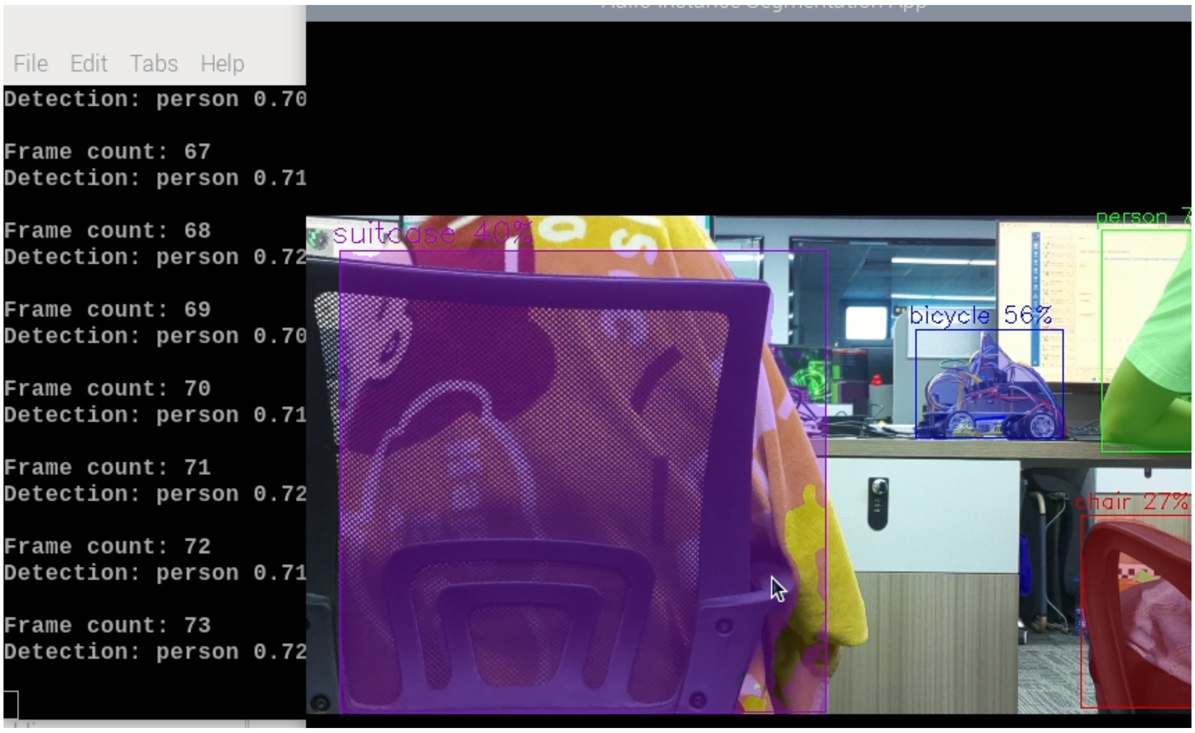
The Callback function shows how to get instance segmentation metadata from the network output. Each instance is represented as an instance with a mask (object). If the flag is set, the code will parse the masks, resize and reshape them according to the frame coordinates. It will print their shapes to the terminal. It is possible to draw masks on the user buffer, but it is not implemented in this example for performance reasons.

`HAILO_DETECTION``HAILO_CONF_CLASS_MASK``--use-frame`

2. Real-time detection of camera images

In the virtual environment terminal, enter the following command. By default, these examples run with a USB camera (/dev/video0). You can change the input source according to the real-time situation using the --input flag.

```
python basic_pipelines/instance_segmentation.py --input /dev/video0 #usb camera
python basic_pipelines/instance_segmentation.py --input rpi #csi camera
```



```
Detection: person 0.70

Frame count: 67
Detection: person 0.71

Frame count: 68
Detection: person 0.72

Frame count: 69
Detection: person 0.70

Frame count: 70
Detection: person 0.71

Frame count: 71
Detection: person 0.72

Frame count: 72
Detection: person 0.71

Frame count: 73
Detection: person 0.72
```

If the camera does not display properly, you can check which video devices are available by running the following command:

```
ls /dev/video*
```

You can test whether the camera is working properly by running the following command:

```
ffplay -f v4l2 /dev/video0
```

If you encounter an error, try another device, such as /dev/video2.

Troubleshooting and known issues

If the camera case suddenly stops running in the middle, you can refer to the following solution steps. If it still doesn't work, you need to restart the Raspberry Pi.

- The RPi camera input is still in Beta. It may be unstable and may cause your application to crash.
- The frame buffer is not optimized and may slow down your application. It is shown as a simple example.
- **DEVICE_IN_USE() error.** This error usually means that the Hailo device (usually) is currently being accessed or locked by another process. This can happen during concurrent access

attempts or if the previous process was not cleanly terminated, leaving the device in a locked state.

Resolution steps:

1.**Identify the device:** Typically, Hailo devices are located at . Make sure this is the correct device file for your setup. `/dev/hailo0`

2.**Find processes using the device:** Run the following command to list any processes currently using the Hailo device:

```
sudo lsof /dev/hailo0
```

3.**Kill the process:** Kill the process using the PID (Process ID) from the output of the previous command. Replace with the actual PID. `<PID>`

```
sudo kill -9 <PID>
```