Instance segmentation example

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

1. Detect the video

Terminal input,

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

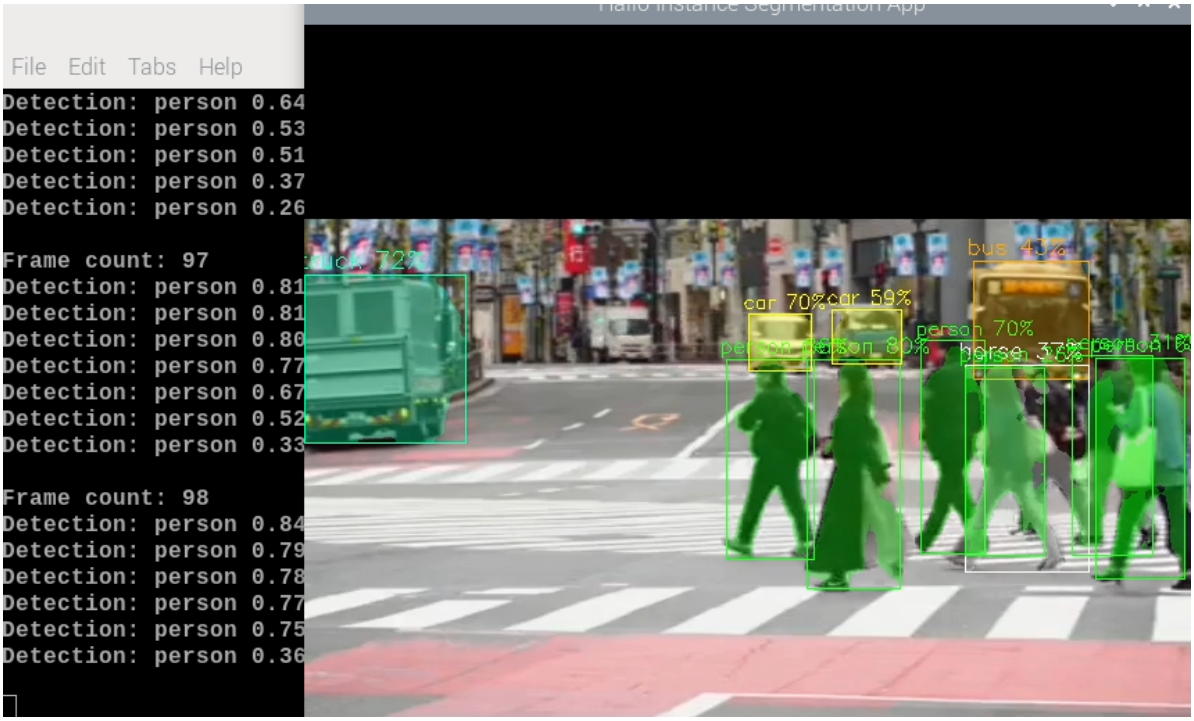
Then enter the command to enter the virtual environment

```
source setup_env.sh
```

Enter the command. Run the video to detect

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

The .mp4 suffix indicates the path of the video



To close the appearance, press ctrl+c.

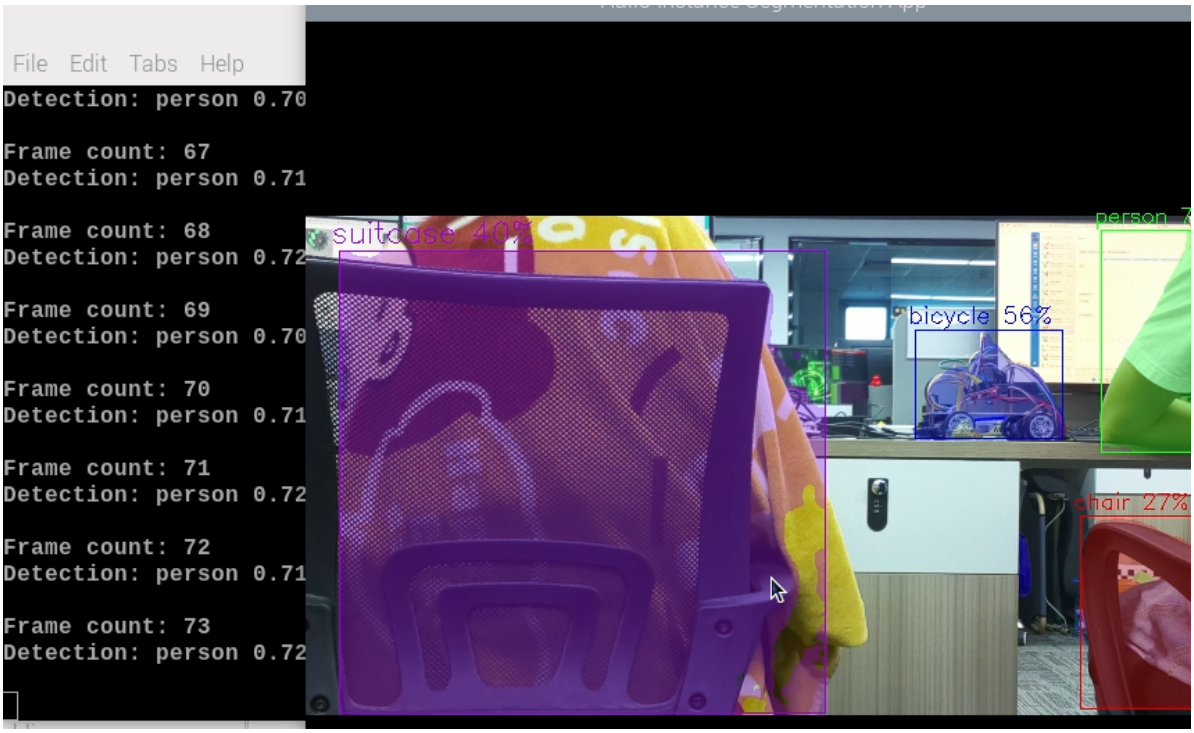
What's in this example:

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 flag is set, the code will parse the masks, resize and reshape them according to frame coordinates. It will print their shape to the terminal. It is possible to draw the 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 camera image detection

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

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



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 that the camera is working properly by running the following command:

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

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

Troubleshooting and known issues

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

- RPi camera input is still in Beta. It may be unstable and may cause application crashes.
- Frame buffer is not optimized and may slow down the application. It is shown as a simple example.
- **DEVICE_IN_USE() error.** This error usually means that the Hailo device is (usually) 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.

Solution Steps:

1.**Identify the device:** Typically, the Hailo device is 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.**Terminate the process:** Use the PID (Process ID) from the previous command output to terminate the process. Replace with the actual PID. `<PID>`

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