Image segmentation mobilenet_unet

1.Introduction

The mobilenet_unet segmentation algorithm example uses images as input and performs algorithm inference using BPU. It publishes segmentation result messages.

The mobilenet_unet model is trained on the Cityscapes dataset and the Onnx model. It supports segmentation of categories such as humans, vehicles, road surfaces, and road signs.

Code repository: (https://github.com/D-Robotics/hobot_dnn)

Applications: mobilenet_unet, composed of MobileNet and UNet, can segment images at the pixel level. It can be used for road recognition, remote sensing map analysis, medical image diagnosis, and other functions. It is mainly applied in the fields of autonomous driving, geological detection, and medical image analysis.

Background blurring example: https://github.com/rusito-23/mobile_unet_segmentation

2.Preparation

- The RDK platform has been flashed with the provided Ubuntu 20.04/22.04 system image.
- TogetheROS.Bot has been successfully installed on the RDK platform.
- A MIPI or USB camera has been installed on the RDK platform. If there is no camera available, the algorithm's effects can be experienced by using local JPEG/PNG images offline.

3.Usage

3.1Using a USB camera

The mobilenet_unet segmentation example subscribes to images published by the sensor package. After inference, it publishes algorithm messages and saves the rendered images automatically in the running directory. The saved images are named in the format of `render_frameid_timestampInSeconds_timestampInNanoseconds.jpg`.

- Enable USB camera node

```
# Configure the tros.b environment
source /opt/tros/humble/setup.bash
```

```
# Configuring USB camera
export CAM_TYPE=usb

# Start the launch file
ros2 launch dnn_node_example dnn_node_example.launch.py
dnn_example_dump_render_img:=1
dnn_example_config_file:=config/mobilenet_unet_workconfig.json
dnn_example_image_width:=1920 dnn_example_image_height:=1080
```

- Output Results

The terminal output during execution shows the following information:

```
[example-3] [WARN] [1655095719.035374293] [example]: Create ai msg publisher with
topic_name: hobot_dnn_detection
[example-3] [WARN] [1655095719.035493746] [example]: Create img
hbmem_subscription with topic_name: /hbmem_img
[example-3] [WARN] [1655095720.693716453] [img_sub]: Sub img fps 6.85
[example-3] [WARN] [1655095721.072909861] [example]: Smart fps 5.85
[example-3] [WARN] [1655095721.702680885] [img_sub]: Sub img fps 3.97
[example-3] [WARN] [1655095722.486407545] [example]: Smart fps 3.54
[example-3] [WARN] [1655095722.733431396] [img_sub]: Sub img fps 4.85
[example-3] [WARN] [1655095723.888407681] [example]: Smart fps 4.28
[example-3] [WARN] [1655095724.069835983] [img_sub]: Sub img fps 3.74
[example-3] [WARN] [1655095724.900725522] [example]: Smart fps 3.95
[example-3] [WARN] [1655095725.093525634] [img_sub]: Sub img fps 3.91
```

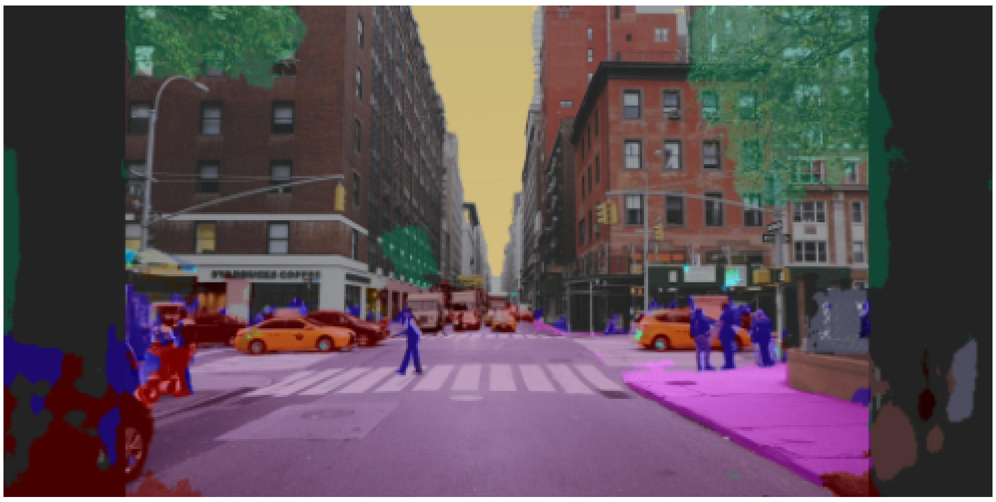
Friendly Reminder:

You must enter the above information in the terminal.

The log output shows that the topic used for publishing the algorithm inference results is `hobot_dnn_detection`, and the topic used for subscribing to the images is `/hbmem_img`. The frame rate at which the images are published will adapt according to the algorithm inference output frame rate. Additionally, rendering the semantic segmentation results on the RDK and saving the images in the running path will cause a decrease in frame rate.

Original image:

Rendered image:



3.2Using IMX219 camera

The mobilenet_unet segmentation example subscribes to images published by the sensor package. After inference, it publishes algorithm messages and saves the rendered images automatically in the running directory. The saved images are named in the format of `render_frameid_timestampInSeconds_timestampInNanoseconds.jpg`.

- Enable MIPI camera node

```
# Configure the tros.b environment
source /opt/tros/humble/setup.bash
```

```
# Configuring MIPI camera
export CAM_TYPE=mipi

# Start the launch file
ros2 launch dnn_node_example dnn_node_example.launch.py
dnn_example_dump_render_img:=1
dnn_example_config_file:=config/mobilenet_unet_workconfig.json
dnn_example_image_width:=1920 dnn_example_image_height:=1080
```

- Output Results

The terminal output during execution shows the following information:

```
[example-3] [WARN] [1655095719.035374293] [example]: Create ai msg publisher with
topic_name: hobot_dnn_detection
[example-3] [WARN] [1655095719.035493746] [example]: Create img
hbmem_subscription with topic_name: /hbmem_img
[example-3] [WARN] [1655095720.693716453] [img_sub]: Sub img fps 6.85
[example-3] [WARN] [1655095721.072909861] [example]: Smart fps 5.85
[example-3] [WARN] [1655095721.702680885] [img_sub]: Sub img fps 3.97
[example-3] [WARN] [1655095722.486407545] [example]: Smart fps 3.54
[example-3] [WARN] [1655095722.733431396] [img_sub]: Sub img fps 4.85
[example-3] [WARN] [1655095723.888407681] [example]: Smart fps 4.28
[example-3] [WARN] [1655095724.069835983] [img_sub]: Sub img fps 3.74
[example-3] [WARN] [1655095724.900725522] [example]: Smart fps 3.95
[example-3] [WARN] [1655095725.093525634] [img_sub]: Sub img fps 3.91
```

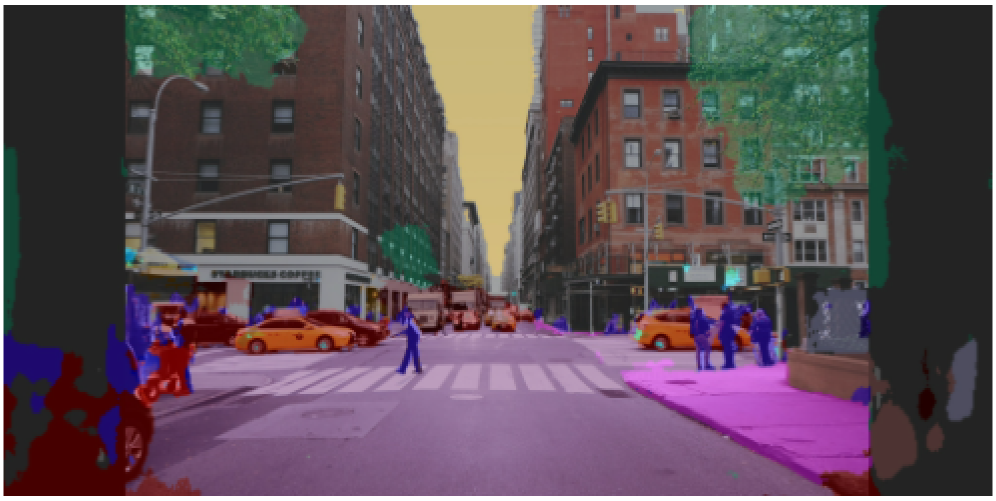
Friendly Reminder:

You must enter the above information in the terminal.

The log output shows that the topic used for publishing the algorithm inference results is `hobot_dnn_detection`, and the topic used for subscribing to the images is `/hbmem_img`. The frame rate at which the images are published will adapt according to the algorithm inference output frame rate. Additionally, rendering the semantic segmentation results on the RDK and saving the images in the running path will cause a decrease in frame rate.

Original image:

Rendered image:



3.3Local image reflow

The mobilenet_unet segmentation example uses local JPEG/PNG format images for feedback. After inference, the rendered images of the algorithm results are stored in the local running path.

- Running the recharge node

```
# Configure the tros.b environment
source /opt/tros/humble/setup.bash
```

```
# Start the launch file
ros2 launch dnn_node_example dnn_node_example_feedback.launch.py
dnn_example_config_file:=config/mobilenet_unet_workconfig.json
dnn_example_image:=config/raw_unet.jpg
```

- Recharge results

The following information is outputted in the terminal:

```
[example-1] [INFO] [1654769881.171005839] [dnn]: The model input 0 width is 2048
and height is 1024
[example-1] [INFO] [1654769881.171129709] [example]: Set output parser.
[example-1] [INFO] [1654769881.171206707] [UnetPostProcess]: Set out parser
[example-1] [INFO] [1654769881.171272663] [dnn]: Task init.
[example-1] [INFO] [1654769881.173427170] [dnn]: Set task_num [2]
[example-1] [INFO] [1654769881.173587414] [example]: The model input width is
2048 and height is 1024
[example-1] [INFO] [1654769881.173646870] [example]: Dnn node feed with local
image: config/raw_unet.jpeg
[example-1] [INFO] [1654769881.750748126] [example]: task_num: 2
[example-1] [INFO] [1654769881.933418736] [example]: Output from image_name:
config/raw_unet.jpeg, frame_id: feedback, stamp: 0.0
[example-1] [INFO] [1654769881.933542440] [UnetPostProcess]: outputs size: 1
[example-1] [INFO] [1654769881.995920396] [UnetPostProcess]: Draw result to file:
render_unet_feedback_0_0.jpeg
```

The log shows that the algorithm performs inference using the input image `config/raw_unet.jpeg`, and the rendered image is stored with the file `render_unet_feedback_0_0.jpeg`. The rendered image looks like this: