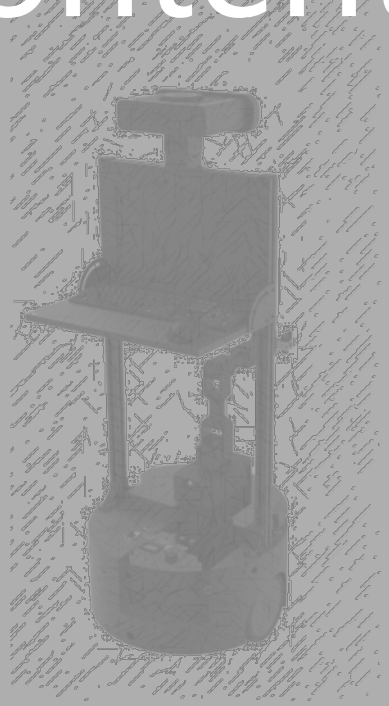
人工智能边缘端应用开发

人脸识别程序开发

曾金恺
广州威控机器人研发工程师

目录
contents



3.1 启动OpenVINO容器

1. 验证容器镜像：
   打开终端，运行命令：

$ xhost +
$ docker images

```
vkrobot@vkrobot:~$ docker images
REPOSITORY              TAG       IMAGE ID       CREATED         SIZE
vkaibot_openvino2021    latest    8ec1514d27fd   4 weeks ago     9.65GB
openvino/workbench      latest    3ecda0d86f11   2 months ago    7.61GB
```

2. 启动OpenVINO容器：

$ docker run --name vkaibot_openvino2021 --net=host -it --rm
-v /tmp/.X11-unix:/tmp/.X11-unix -e DISPLAY=unix$DISPLAY -e GDK_SCALE -e GDK_DPI_SCALE
--privileged -v /dev:/dev
-v ~/vkaibot_ncs/ov_workspace:/home/openvino/vkaibot_ncs/ov_workspace
vkaibot_openvino2021

3.2 启动OpenVINO容器 （续）

```
$ docker run --name vkaibot_openvino2021 --net=host -it --rm
        -v /tmp/.X11-unix:/tmp/.X11-unix -e DISPLAY=unix$DISPLAY -e GDK_SCALE -e GDK_DPI_SCALE
        --privileged -v /dev:/dev
        -v ~/vkaibot_ncs/ov_workspace:/home/openvino/vkaibot_ncs/ov_workspace
         vkaibot_openvino2021
```

- **docker run：** 创建docker容器
- **--name：** 容器名称
- **--net：** 桥接网络
- **-i：** 使用标准输入输出
- **-t：** 创建终端（Terminal）

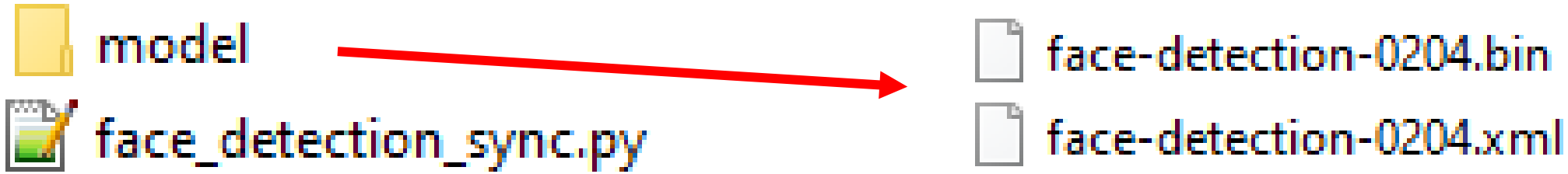
} 通常以 **-it / -ti** 形式连用

- **--rm：** 退出容器时删除
- **--privilege：** 允许容器映射文件夹
- **-v：** 映射（文件夹，硬件设备……） 示例： **-v** /宿主机文件夹路径：/容器内文件夹路径
- **vkaibot_openvino2021：** 镜像名称
- **-v /tmp/.X11-unix:/tmp/.X11-unix -e DISPLAY=unix$DISPLAY -e GDK_SCALE -e GDK_DPI_SCALE：**
  映射显示屏，使得容器内程序可以调用宿主机显示屏输出

```
vkrobot@vkrobot:~/vkaibot_openvino$ ./docker_run_openvino.sh
[setupvars.sh] OpenVINO environment initialized
root@vkrobot:/opt/intel/openvino_2021.2.185#
```

3.3 人脸识别程序

- 人脸识别程序（face_detection）工作目录:
  ~/vkaibot_ncs/ov_workspace/face_detection/
- 人脸识别程序能正确识别人脸并且VKAIBOT显示屏有视频输出
- 能够修改人脸识别程序的主程序，添加额外信息，如标签、置信度、识别延迟等



3.4 人脸识别程序

- face-detection-0204

- 基于： MobileNetV2
- 训练： SSD
- 输入： 1x3x448x448
- 输出： [1, 1, N, 7]

Specification

METRIC	VALUE
AP (WIDER)	92.89%
GFlops	2.406
MParams	1.851
Source framework	PyTorch*

3.5 运行人脸识别程序

- 打开终端，进入工作目录：

  $ export DISPLAY=:0

  $ cd /home/openvino/vkaibot_ncs/ov_workspace/face_detection/

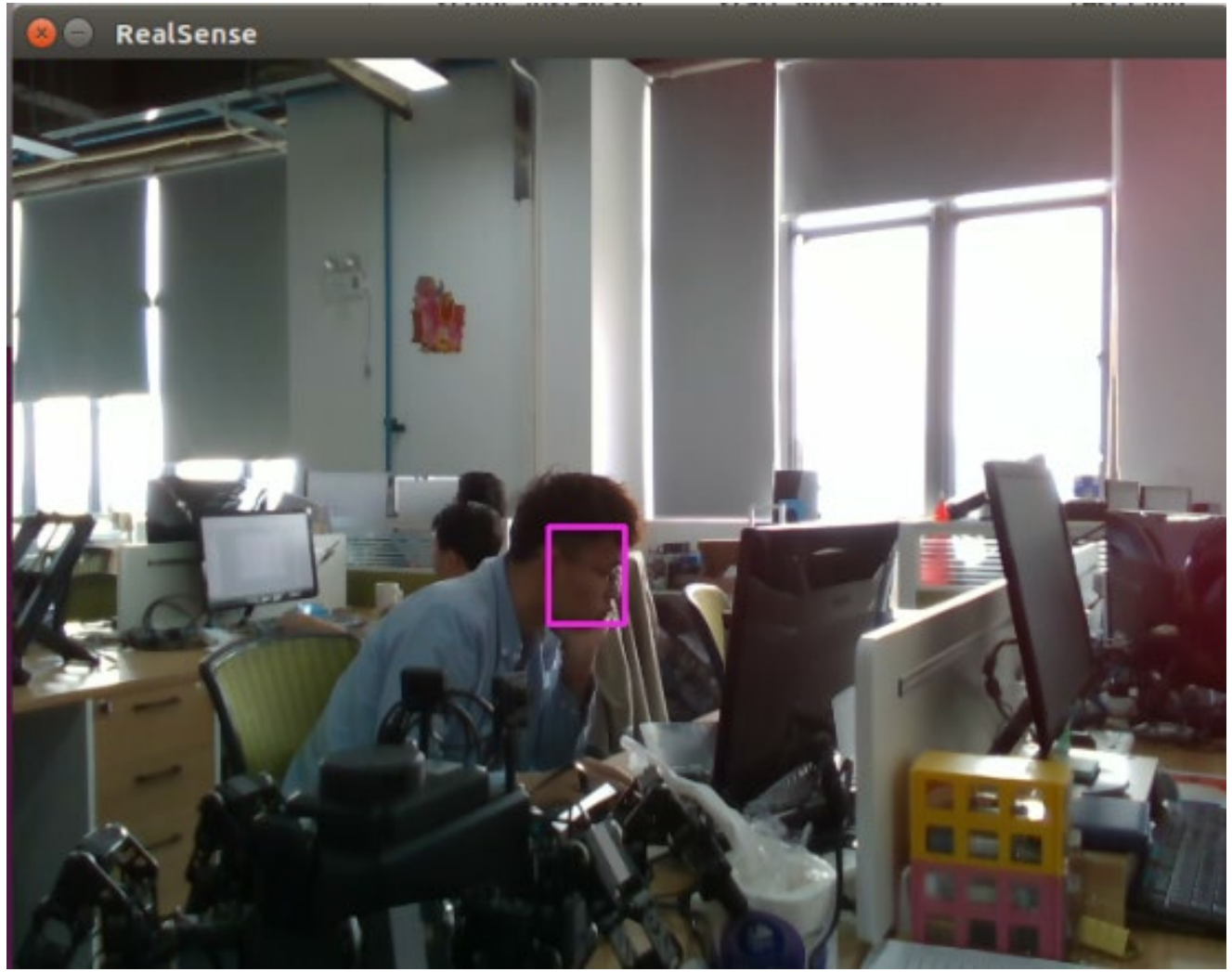
- 运行主程序：

  $ python face_detection_sync.py -m ./model/face-detection-0204.xml

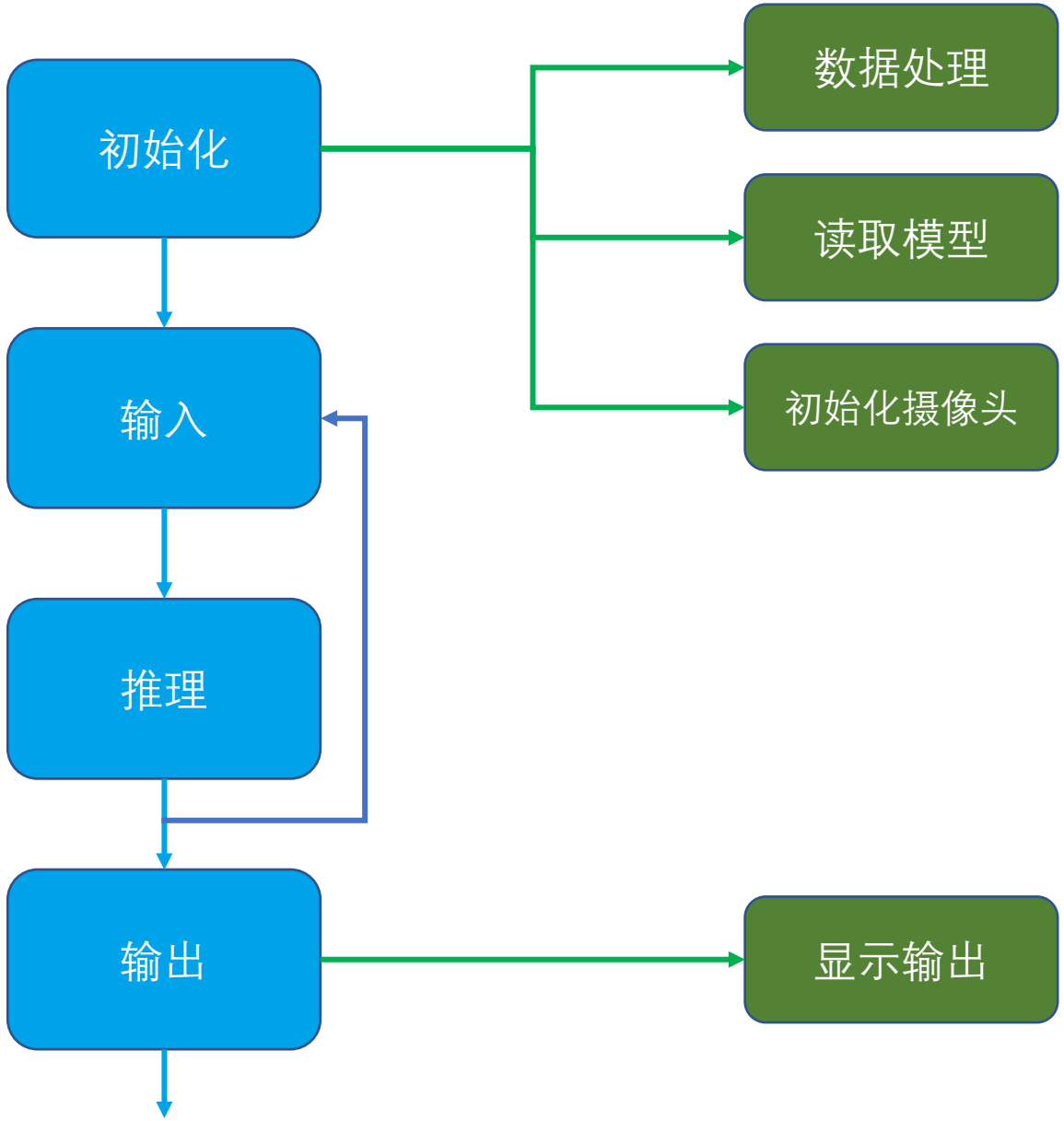
- 查看运行帮助：

  $ python face_detection_sync.py –h

3.5 运行结果



3.6 face_detection_sync.py



3.6 pyrealsense2

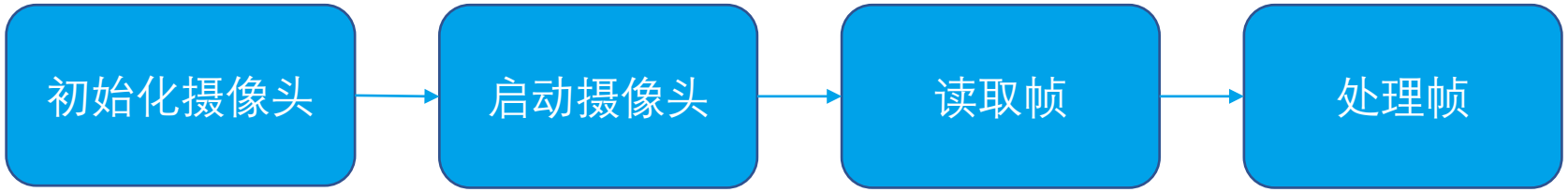
- Python
- 调用Intel RealSense深度摄像头

- 初始化摄像头
- 启动摄像头
- 读取帧

```
# RS Config
pipeline = rs.pipeline()
config = rs.config()
config.enable_stream(rs.stream.depth, 640, 480, rs.format.z16, 30)
config.enable_stream(rs.stream.color, 640, 480, rs.format.bgr8, 30)
# Start streaming
pipeline.start(config)
log.info("Start streaming")
# Read frame
frames = pipeline.wait_for_frames()
color_frame = frames.get_color_frame()
color_image = np.asanyarray(color_frame.get_data())
```

- 处理变量：*color_image*

调用Intel RealSense深度摄像头

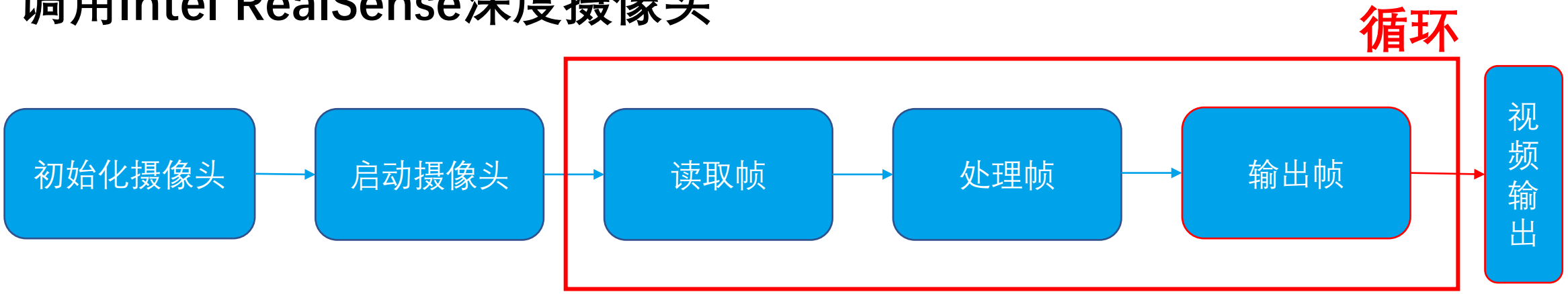


- 初始化摄像头
- 启动摄像头
- 读取帧

```
# RS Config
pipeline = rs.pipeline()
config = rs.config()
config.enable_stream(rs.stream.depth, 640, 480, rs.format.z16, 30)
config.enable_stream(rs.stream.color, 640, 480, rs.format.bgr8, 30)
# Start streaming
pipeline.start(config)
log.info("Start streaming")
# Read frame
frames = pipeline.wait_for_frames()
color_frame = frames.get_color_frame()
color_image = np.asanyarray(color_frame.get_data())
```

3.7 运行过程

调用Intel RealSense深度摄像头

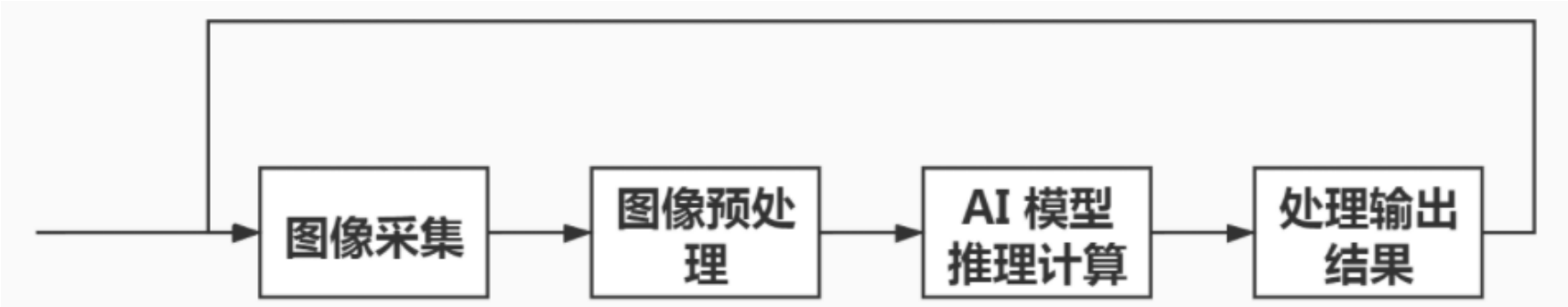


- 初始化摄像头
- 启动摄像头
- 读取帧

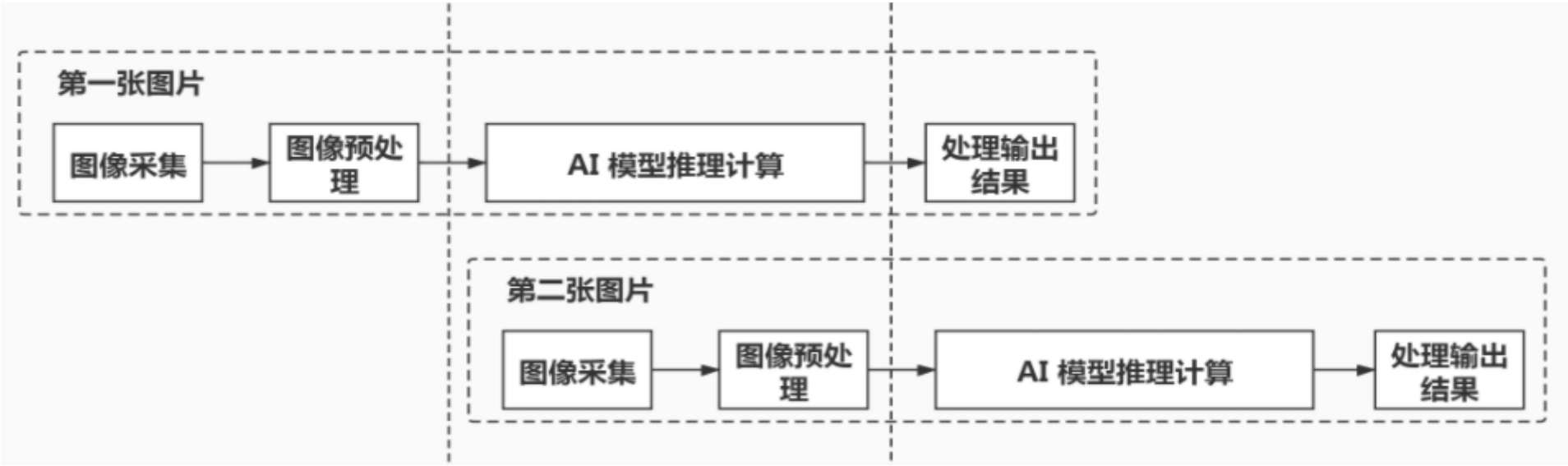
```
# RS Config
pipeline = rs.pipeline()
config = rs.config()
config.enable_stream(rs.stream.depth, 640, 480, rs.format.z16, 30)
config.enable_stream(rs.stream.color, 640, 480, rs.format.bgr8, 30)
# Start streaming
pipeline.start(config)
log.info("Start streaming")
# Read frame
frames = pipeline.wait_for_frames()
color_frame = frames.get_color_frame()
color_image = np.asanyarray(color_frame.get_data())
```

3.7 运行过程

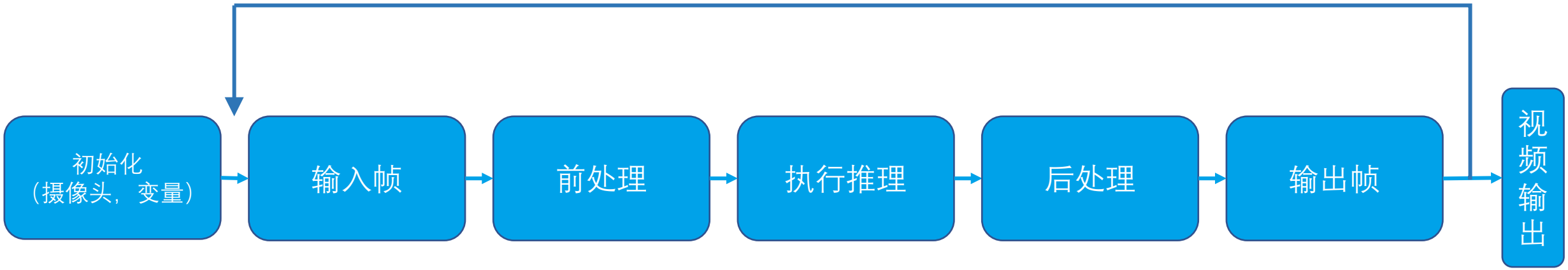
- 同步模式：



- 异步模式：



3.8 程序流程



3.9 程序分析

- 配置变量

```
31  def build_argparser():
32      parser = ArgumentParser(add_help=False)
33      args = parser.add_argument_group("Options")
34      args.add_argument('-h', '--help', action='help', default=SUPPRESS, help='Show this help message and exit.')
35      args.add_argument("-m", "--model", help="Required. Path to an .xml or .onnx file with a trained model.",
36                        required=True, type=str)
37      args.add_argument("-l", "--cpu_extension",
38                        help="Optional. Required for CPU custom layers. "
39                             "Absolute path to a shared library with the kernels implementations.",
40                        type=str, default=None)
41      args.add_argument("-d", "--device",
42                        help="Optional. Specify the target device to infer on; "
43                             "CPU, GPU, FPGA or MYRIAD is acceptable. "
44                             "Sample will look for a suitable plugin for device specified (CPU by default)",
45                        default="CPU", type=str)
46      args.add_argument("--labels", help="Optional. Labels mapping file", default=None, type=str)
47      args.add_argument("-nt", "--number_top", help="Optional. Number of top results", default=10, type=int)
48
49      return parser
50
```

3.9 程序分析

- Main 函数：
  - 读取参数
  - 读取模型
  - 初始化图片变量
  - 载入模型网络

```python
def main():
    log.basicConfig(format="[ %(levelname)s ] %(message)s", level=log.INFO, stream=sys.stdout)
    args = build_argparser().parse_args()
    log.info("Loading Inference Engine")
    ie = IECore()

    # ---1. Read a model in OpenVINO Intermediate Representation (.xml and .bin files) ---
    model = args.model
    log.info(f"Loading network:\n\t{model}")
    net = ie.read_network(model=model)
    func = ng.function_from_cnn(net)
    ops = func.get_ordered_ops()
    # -----------------------------------------------------------------------------------------------------

    # ------------- 2. Load Plugin for inference engine and extensions library if specified ----
    log.info("Device info:")
    versions = ie.get_versions(args.device)
    print("{}{}".format(" " * 8, args.device))
    print("{}MKLDNNPlugin version ......... {}.{}".format(" " * 8, versions[args.device].major,
                                                          versions[args.device].minor))
    print("{}Build ........... {}".format(" " * 8, versions[args.device].build_number))

    if args.cpu_extension and "CPU" in args.device:
        ie.add_extension(args.cpu_extension, "CPU")
        log.info("CPU extension loaded: {}".format(args.cpu_extension))
    # -----------------------------------------------------------------------------------------------------

    # --------------------------- 3. Read and preprocess input --------------------------------------

    for input_key in net.input_info:
        if len(net.input_info[input_key].input_data.layout) == 4:
            n, c, h, w = net.input_info[input_key].input_data.shape

    images = np.ndarray(shape=(n, c, h, w))
    images_hw = []
    exec_net = ie.load_network(network=net, device_name=args.device)
```

3.9 程序分析

- 配置RealSense摄像头参数
- 启动RealSense摄像头

```
89      # RS Config
90      pipeline = rs.pipeline()
91      config = rs.config()
92      config.enable_stream(rs.stream.depth, 640, 480, rs.format.z16, 30)
93      config.enable_stream(rs.stream.color, 640, 480, rs.format.bgr8, 30)
94      # Start streaming
95      pipeline.start(config)
96      log.info("Start streaming")
```

3.9 程序分析

- loop：
  - 读取帧
  - 调整图片大小以及维度（与输入数据相同）

```
97      try:
98          while True:
99              # Read frame
100             frames = pipeline.wait_for_frames()
101             color_frame = frames.get_color_frame()
102             color_image = np.asanyarray(color_frame.get_data())
103             for i in range(n):
104                 image = color_image
105                 ih, iw = image.shape[:-1]
106                 images_hw.append((ih, iw))
107                 if (ih, iw) != (h, w):
108                     image = cv2.resize(image, (w, h))
109                 image = image.transpose((2, 0, 1))  # Change data layout from HWC to CHW
110                 images[i] = image
```

3.9 程序分析

- 准备输入数据
- 准备输出数据

```
115     # ---------------------------- 4. Configure input & output ----------------------------------------------
116     # ---------------------------- Prepare input blobs -----------------------------------------------------
117         assert (len(net.input_info.keys()) == 1 or len(
118             net.input_info.keys()) == 2), "Sample supports topologies only with 1 or 2 inputs"
119         out_blob = next(iter(net.outputs))
120         input_name, input_info_name = "", ""
121
122         for input_key in net.input_info:
123             if len(net.input_info[input_key].layout) == 4:
124                 input_name = input_key
125                 net.input_info[input_key].precision = 'U8'
126             elif len(net.input_info[input_key].layout) == 2:
127                 input_info_name = input_key
128                 net.input_info[input_key].precision = 'FP32'
129
130         data = {}
131         data[input_name] = images
132
133         if input_info_name != "":
134             infos = np.ndarray(shape=(n, c), dtype=float)
135             for i in range(n):
136                 infos[i, 0] = h
137                 infos[i, 1] = w
138                 infos[i, 2] = 1.0
139             data[input_info_name] = infos
140
141     # ---------------------------- Prepare output blobs ----------------------------------------------------
142         #log.info('Preparing output blobs')
143         output_name, output_info = "", net.outputs[next(iter(net.outputs.keys()))]
144         output_ops = {op.friendly_name : op for op in ops \
145                       if op.friendly_name in net.outputs and op.get_type_name() == "DetectionOutput"}
146
147         output_dims = output_info.shape
148
149         max_proposal_count, object_size = output_dims[2], output_dims[3]
150
151         output_info.precision = "FP32"
```

- 执行推理

```
154     # ---------------------------- Performing inference -----------
155             res = exec_net.infer(inputs=data)
156     # -------------------------------------------------------------
```

3.9 程序分析

- 解码输出数据
- 后处理

```
158     # ------------------------- Read and postprocess output -----------------
159             res = res[out_blob]
160             boxes, classes = {}, {}
161             data = res[0][0]
162             for number, proposal in enumerate(data):
163                 if proposal[2] > 0:
164                     imid = np.int(proposal[0])
165                     ih, iw = images_hw[imid]
166                     label = np.int(proposal[1])
167                     confidence = proposal[2]
168                     xmin = np.int(iw * proposal[3])
169                     ymin = np.int(ih * proposal[4])
170                     xmax = np.int(iw * proposal[5])
171                     ymax = np.int(ih * proposal[6])
172                     if proposal[2] > 0.5:
173                         if not imid in boxes.keys():
174                             boxes[imid] = []
175                         boxes[imid].append([xmin, ymin, xmax, ymax])
176                         if not imid in classes.keys():
177                             classes[imid] = []
178                         classes[imid].append(label)
```

3.9 程序分析

- 输出帧
- 添加识别框
- 输出处理完成的帧 → 回到读取新的一帧

```
180                 # Output
181                 tmp_image = color_image
182                 try:
183                     for box in boxes[imid]:
184                         cv2.rectangle(tmp_image, (box[0], box[1]), (box[2], box[3]), (232, 35, 244), 2)
185                 except:
186                     pass
187                 cv2.namedWindow('RealSense', cv2.WINDOW_AUTOSIZE)
188                 cv2.imshow('RealSense', tmp_image)
189                 key = cv2.waitKey(1)
190                 # Press esc or 'q' to close the image window
191                 if key & 0xFF == ord('q') or key == 27:
192                     cv2.destroyAllWindows()
193                     break
194     # ------------------------------------------------------------------------------------------
```

3.10 练习

1. 修改人脸识别程序，更改识别框颜色为：绿色
   - cv2中RGB色彩格式为 BGR
2. 修改人脸识别程序，为识别框添加表示：置信度
   - cv2.putText(图片，内容，位置，cv2.FONT_HERSHEY_SIMPLEX，0.8，颜色)
     字体　　笔画粗细
3. 视频输出窗口中应能够显示：神经网络推理时间（ms）
   - cv2.putText()
   - time 函数

VIK ROBOT

Thanks