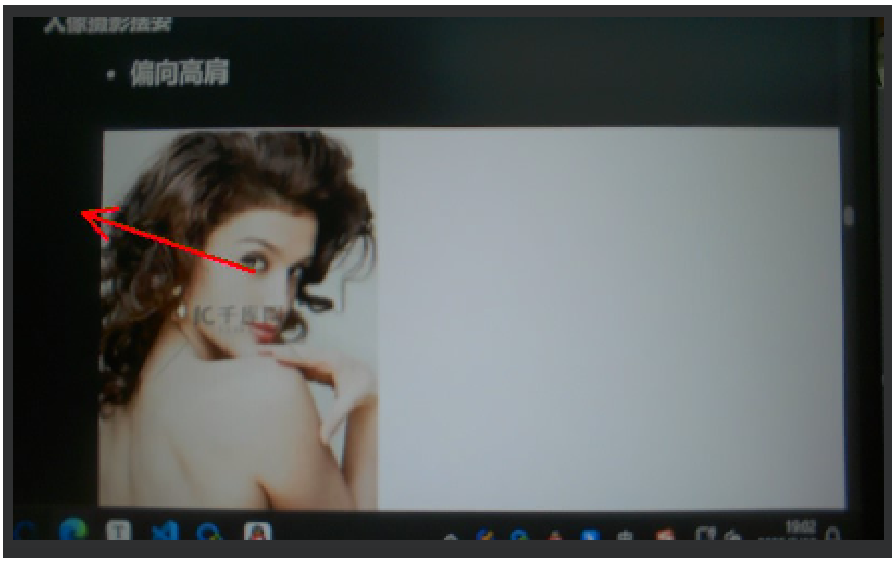
Gaze direction detection

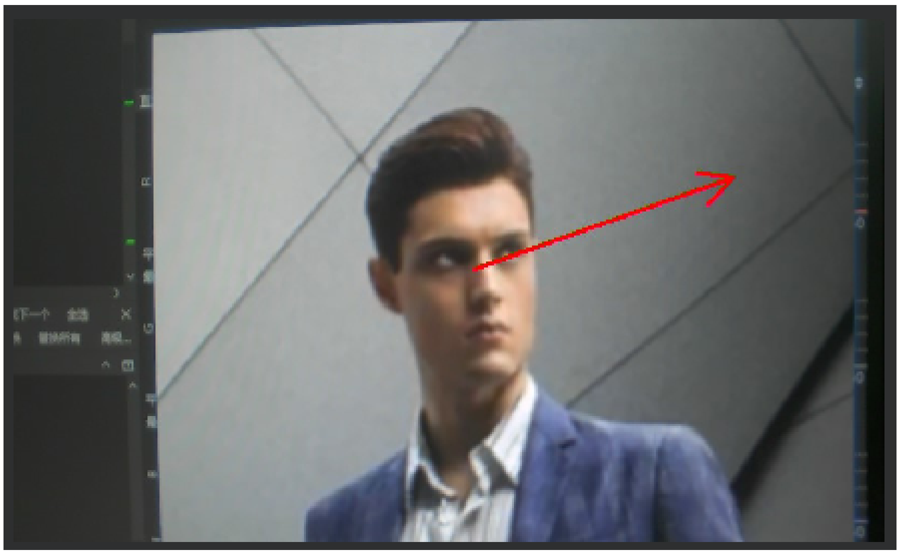
Routine Experiment Effect

In this section, we will demonstrate how to use K230 to detect and judge the direction of eye gaze.

The code in this section is in [Source code/07.Face/05.eye_gaze.py]

After connecting to the IDE, run the example code in this section and aim the K230 at the face. You can see the direction the eyes are currently looking at will be marked on the screen.





> Serial port output function has been added
>
> After detecting a face and recognizing the gaze direction, the following serial output format will be sent
>
> $x0,y0,x1,y1#
>
> The '$' represents the beginning of the data, and the '#' represents the end of the data.
>
> x0, y0, x1, y1 represent the arrows that identify the direction, where (x0, y0) are the starting coordinates of the arrow, and (x1, y1) are the ending coordinates of the arrow.

Code Explanation

The procedure is mainly divided into three steps:

1. Video Capture
    - Get the real-time camera image through Pipeline
    - One channel is used for AI reasoning, and the other channel is output to the display/IDE
2. Two-stage visual processing
    - Phase 1: Face Detection • Use face_detection_320.kmodel to locate the face region • Input size: 320x320 • Optimize the detection results by confidence_threshold(0.5) and nms_threshold(0.2)
    - Second stage: Gaze direction estimation • Use eye_gaze.kmodel to analyze the extracted face region • Input size: 448x448 • Output parameters: pitch and yaw
3. Results Visualization
    - Based on pitch (up and down) and yaw (left and right)
    - Convert an angle to a 2D vector using trigonometric functions
    - Draw a red arrow on the screen to indicate the gaze direction

Code structure

Import related dependency classes

```
from  libs . PipeLine  import  PipeLine , ScopedTiming
from  libs .AIBase import AIBase
from  libs . AI2D  import  Ai2d
import  os
import  ujson
from  media . media  import  *
from  time  import  *
import  nncase_runtime  as  nn
import  ulab . numpy  as  np
import  time
import  image
import  aidemo
import  random
import  gc
import  sys
import  math

e.g. = None
```

Custom face detection task class

```
class FaceDetApp(AIBase):
    def
__init__(self,kmodel_path,model_input_size,anchors,confidence_threshold=0.25,nms
_threshold=0.3,rgb888p_size=[1280,720],display_size=[640,360],debug_mode=0):
        super().__init__(kmodel_path,model_input_size,rgb888p_size,debug_mode)
        # Path to the kmodel file
        # kmodel路径
        self.kmodel_path=kmodel_path
        # Input resolution of the detection model
        # 检测模型输入分辨率
        self.model_input_size=model_input_size
        # Confidence threshold for detection
        # 置信度阈值
        self.confidence_threshold=confidence_threshold
        # NMS threshold for detection
        # nms阈值
        self.nms_threshold=nms_threshold
        self.anchors=anchors
        # Sensor input resolution, aligned to 16 bytes
        # sensor给到AI的图像分辨率，宽16字节对齐
        self.rgb888p_size=[ALIGN_UP(rgb888p_size[0],16),rgb888p_size[1]]
        # Display resolution, aligned to 16 bytes
        # 视频输出VO分辨率，宽16字节对齐
        self.display_size=[ALIGN_UP(display_size[0],16),display_size[1]]
        # Debug mode
        # debug模式
        self.debug_mode=debug_mode
        # Ai2d instance for preprocessing
        # Ai2d实例，用于实现模型预处理
        self.ai2d=Ai2d(debug_mode)
        # Set the input/output format and data type for Ai2d
        # 设置Ai2d的输入输出格式和类型

 self.ai2d.set_ai2d_dtype(nn.ai2d_format.NCHW_FMT,nn.ai2d_format.NCHW_FMT,np.uin
t8, np.uint8)

    # Configure the preprocessing operations, using padding and resize
    # Ai2d supports crop/shift/pad/resize/affine, check the code in
/sdcard/app/libs/AI2D.py for details
    # 配置预处理操作，这里使用了padding和resize，Ai2d支持crop/shift/pad/resize/affine，
具体代码请打开/sdcard/app/libs/AI2D.py查看
    def config_preprocess(self,input_image_size=None):
        with ScopedTiming("set preprocess config",self.debug_mode > 0):
            # Initialize the Ai2d preprocessing config, use the sensor
resolution by default, or set input_image_size to customize
            # 初始化ai2d预处理配置，默认为sensor给到AI的尺寸，可以通过设置
input_image_size自行修改输入尺寸
            ai2d_input_size=input_image_size if input_image_size else
self.rgb888p_size
            # Set the padding preprocessing
            # 设置padding预处理
            self.ai2d.pad(self.get_pad_param(), 0, [104,117,123])
            # Set the resize preprocessing
            # 设置resize预处理
            self.ai2d.resize(nn.interp_method.tf_bilinear,
nn.interp_mode.half_pixel)
            # Build the preprocessing pipeline, with the input and output tensor
shapes
            # 构建预处理流程,参数为预处理输入tensor的shape和预处理输出的tensor的shape
            self.ai2d.build([1,3,ai2d_input_size[1],ai2d_input_size[0]],
[1,3,self.model_input_size[1],self.model_input_size[0]])

    # Custom postprocessing, using the face_det_post_process interface from the
aidemo library
    # 自定义任务后处理,这里使用了aidemo库的face_det_post_process接口
    def postprocess(self,results):
        with ScopedTiming("postprocess",self.debug_mode > 0):
            res =
aidemo.face_det_post_process(self.confidence_threshold,self.nms_threshold,self.m
odel_input_size[0],self.anchors,self.rgb888p_size,results)
            if len(res)==0:
                return res
            else:
                return res[0]

    # Calculate the padding parameters
    # 计算padding参数
    def get_pad_param(self):
        dst_w = self.model_input_size[0]
        dst_h = self.model_input_size[1]
        # Calculate the minimum scaling ratio, keep the aspect ratio
        # 计算最小的缩放比例，等比例缩放
        ratio_w = dst_w / self.rgb888p_size[0]
        ratio_h = dst_h / self.rgb888p_size[1]
        if ratio_w < ratio_h:
            ratio = ratio_w
        else:
            ratio = ratio_h
        new_w = (int)(ratio * self.rgb888p_size[0])
        new_h = (int)(ratio * self.rgb888p_size[1])
        dw = (dst_w - new_w) / 2
        dh = (dst_h - new_h) / 2
        top = (int)(round(0))
        bottom = (int)(round(dh * 2 + 0.1))
        left = (int)(round(0))
        right = (int)(round(dw * 2 - 0.1))
        return [0,0,0,0,top, bottom, left, right]
```

Customizing the Gaze Estimation Task Class

```
# Custom Eye Gaze Estimation Application
# 自定义注视估计任务类
class EyeGazeApp(AIBase):
    def __init__(self,kmodel_path,model_input_size,rgb888p_size=
[640,360],display_size=[640,360],debug_mode=0):
        super().__init__(kmodel_path,model_input_size,rgb888p_size,debug_mode)
        # Path to the kmodel file
        # kmodel路径
        self.kmodel_path=kmodel_path
        # Input resolution of the gaze estimation model
        # 注视估计模型输入分辨率
        self.model_input_size=model_input_size
        # Sensor input resolution, aligned to 16 bytes
        # sensor给到AI的图像分辨率，宽16字节对齐
        self.rgb888p_size=[ALIGN_UP(rgb888p_size[0],16),rgb888p_size[1]]
        # Display resolution, aligned to 16 bytes
        # 视频输出VO分辨率，宽16字节对齐
        self.display_size=[ALIGN_UP(display_size[0],16),display_size[1]]
        # Debug mode
        # debug模式
        self.debug_mode=debug_mode
        # Ai2d instance for preprocessing
        # Ai2d实例，用于实现模型预处理
        self.ai2d=Ai2d(debug_mode)
        # Set the input/output format and data type for Ai2d
        # 设置Ai2d的输入输出格式和类型

 self.ai2d.set_ai2d_dtype(nn.ai2d_format.NCHW_FMT,nn.ai2d_format.NCHW_FMT,np.uin
t8, np.uint8)

    # Configure the preprocessing operations, using crop and resize
    # Ai2d supports crop/shift/pad/resize/affine
    # 配置预处理操作，这里使用了crop和resize
    def config_preprocess(self,det,input_image_size=None):
        with ScopedTiming("set preprocess config",self.debug_mode > 0):
            # Initialize the Ai2d preprocessing config
            # 初始化ai2d预处理配置
            ai2d_input_size=input_image_size if input_image_size else
self.rgb888p_size
            # Calculate the crop parameters based on the detection result
            # 计算crop预处理参数
            x, y, w, h = map(lambda x: int(round(x, 0)), det[:4])
            # Set the crop preprocessing
            # 设置crop预处理
            self.ai2d.crop(x,y,w,h)
            # Set the resize preprocessing
            # 设置resize预处理
            self.ai2d.resize(nn.interp_method.tf_bilinear,
nn.interp_mode.half_pixel)
            # Build the preprocessing pipeline, with the input and output tensor
shapes
            # 构建预处理流程,参数为预处理输入tensor的shape和预处理输出的tensor的shape
            self.ai2d.build([1,3,ai2d_input_size[1],ai2d_input_size[0]],
[1,3,self.model_input_size[1],self.model_input_size[0]])
```

```
    # Custom postprocessing, results is a list of output arrays from the model
    # This calls the eye_gaze_post_process interface from the aidemo library
    # 自定义后处理，results是模型输出的array列表，这里调用了aidemo库的
eye_gaze_post_process接口
    def postprocess(self,results):
        with ScopedTiming("postprocess",self.debug_mode > 0):
            post_ret = aidemo.eye_gaze_post_process(results)
            return post_ret[0],post_ret[1]
```

Custom gaze estimation class

```
# Custom Eye Gaze Estimation Class
# 自定义注视估计类
class EyeGaze:
    def
__init__(self,face_det_kmodel,eye_gaze_kmodel,det_input_size,eye_gaze_input_size
,anchors,confidence_threshold=0.25,nms_threshold=0.3,rgb888p_size=
[640,360],display_size=[640,360],debug_mode=0):
        # Path to the face detection kmodel
        # 人脸检测模型路径
        self.face_det_kmodel=face_det_kmodel
        # Path to the eye gaze estimation kmodel
        # 人脸注视估计模型路径
        self.eye_gaze_kmodel=eye_gaze_kmodel
        # Input resolution of the face detection model
        # 人脸检测模型输入分辨率
        self.det_input_size=det_input_size
        # Input resolution of the eye gaze estimation model
        # 人脸注视估计模型输入分辨率
        self.eye_gaze_input_size=eye_gaze_input_size
        # Anchors
        # anchors
        self.anchors=anchors
        # Confidence threshold
        # 置信度阈值
        self.confidence_threshold=confidence_threshold
        # NMS threshold
        # nms阈值
        self.nms_threshold=nms_threshold
        # Sensor input resolution, aligned to 16 bytes
        # sensor给到AI的图像分辨率，宽16字节对齐
        self.rgb888p_size=[ALIGN_UP(rgb888p_size[0],16),rgb888p_size[1]]
        # Display resolution, aligned to 16 bytes
        # 视频输出VO分辨率，宽16字节对齐
        self.display_size=[ALIGN_UP(display_size[0],16),display_size[1]]
        # Debug mode
        # debug_mode模式
        self.debug_mode=debug_mode
        # Face detection instance
        # 人脸检测实例

 self.face_det=FaceDetApp(self.face_det_kmodel,model_input_size=self.det_input_s
ize,anchors=self.anchors,confidence_threshold=self.confidence_threshold,nms_thre
shold=self.nms_threshold,rgb888p_size=self.rgb888p_size,display_size=self.displa
y_size,debug_mode=0)
        # Eye gaze estimation instance
        # 注视估计实例
 self.eye_gaze=EyeGazeApp(self.eye_gaze_kmodel,model_input_size=self.eye_gaze_in
put_size,rgb888p_size=self.rgb888p_size,display_size=self.display_size)
        # Configure the preprocessing for face detection
        # 人脸检测配置预处理
        self.face_det.config_preprocess()

    # Run the pipeline
    # run方法
    def run(self,input_np):
        # First perform face detection
        # 先进行人脸检测
        det_boxes=self.face_det.run(input_np)
        eye_gaze_res=[]
        for det_box in det_boxes:
            # Perform eye gaze estimation for each detected face
            # 对每一个检测到的人脸做注视估计
            self.eye_gaze.config_preprocess(det_box)
            pitch,yaw=self.eye_gaze.run(input_np)
            eye_gaze_res.append((pitch,yaw))
        return det_boxes,eye_gaze_res

    # Draw the eye gaze estimation result
    # 绘制注视估计效果
    def draw_result(self,pl,dets,eye_gaze_res):
        pl.osd_img.clear()
        if dets:
            for det,gaze_ret in zip(dets,eye_gaze_res):
                pitch , yaw = gaze_ret
                length = self.display_size[0]/ 2
                x, y, w, h = map(lambda x: int(round(x, 0)), det[:4])
                x = x * self.display_size[0] // self.rgb888p_size[0]
                y = y * self.display_size[1] // self.rgb888p_size[1]
                w = w * self.display_size[0] // self.rgb888p_size[0]
                h = h * self.display_size[1] // self.rgb888p_size[1]
                center_x = (x + w / 2.0)
                center_y = (y + h / 2.0)
                dx = -length * math.sin(pitch) * math.cos(yaw)
                target_x = int(center_x + dx)
                dy = -length * math.sin(yaw)
                target_y = int(center_y + dy)
                pl.osd_img.draw_arrow(int(center_x), int(center_y), target_x,
target_y, color = (255,255,0,0),
 size = 30, thickness = 4)
```

Execute routine code

```
# 执行演示的主函数 / Main function to execute demonstration
def exce_demo(pl):
    global eg
    display_mode = pl.display_mode
    rgb888p_size = pl.rgb888p_size
    display_size = pl.display_size

    # 设置模型路径 / Set model paths
    face_det_kmodel_path="/sdcard/kmodel/face_detection_320.kmodel"
    # Path to the eye gaze estimation kmodel
    # 人脸注视估计模型路径
    eye_gaze_kmodel_path="/sdcard/kmodel/eye_gaze.kmodel"
    # Other parameters
    # 其他参数
    anchors_path="/sdcard/utils/prior_data_320.bin"

    face_det_input_size=[320,320]
    eye_gaze_input_size=[448,448]
    confidence_threshold=0.5
    nms_threshold=0.2
    anchor_len=4200
    det_dim=4
    anchors = np.fromfile(anchors_path, dtype=np.float)
    anchors = anchors.reshape((anchor_len,det_dim))


 eg=EyeGaze(face_det_kmodel_path,eye_gaze_kmodel_path,det_input_size=face_det_in
put_size,eye_gaze_input_size=eye_gaze_input_size,anchors=anchors,confidence_thre
shold=confidence_threshold,nms_threshold=nms_threshold,rgb888p_size=rgb888p_size
,display_size=display_size)

    # 主循环 / Main loop
    try:
        while True:
            with ScopedTiming("total",1):
                img=pl.get_frame()                      # Get the current
frame
                det_boxes,eye_gaze_res=eg.run(img)      # Run inference on
the current frame
                eg.draw_result(pl,det_boxes,eye_gaze_res)   # Draw the inference
result
                pl.show_image()                         # Display the result
                gc.collect()
    except Exception as e:
        print("人脸注视方向功能退出")                     # 打印退出信息 / Print
exit message
    finally:
        # 清理资源 / Clean up resources
        eg.face_det.deinit()
        eg.eye_gaze.deinit()

# 退出函数 / Exit function
def exit_demo():
    global eg
    # 清理所有资源 / Clean up all resources
    eg.face_det.deinit()
    eg.eye_gaze.deinit()

# 程序入口 / Program entry
if __name__=="__main__":
    # 设置基本参数 / Set basic parameters
    rgb888p_size=[640,360]
    display_size=[640,480]
    display_mode="lcd"

    # 创建并启动视频处理模块 / Create and start video processing pipeline
 pl=PipeLine(rgb888p_size=rgb888p_size,display_size=display_size,display_mode=di
splay_mode)
    pl.create()
    exce_demo(pl)
```