**Procedure: Head Pose Estimation**

In this experiment, you will implement head pose estimation application on the Atlas200 DK. This application can be useful in several scenarios; one such scenario is determining the viewing direction of a driver in an **automated driver assistance system** installed in a vehicle. If the driver is found to be looking away (up/down/left/right) from the optimal viewing direction (straight ahead) for an extended period of time, the system can generate an alert to bring the driver’s attention back, and mitigate risky behaviors.

This method first detects a **face** region located in the given input image and then uses the cropped region of the detected face as the input to the head pose estimation model. The head pose of a person is calculated in terms of 3 angles**: *yaw*, *pitch* and *roll*** in an image. Detailed steps, and explanations are provided in this guide, so you can understand how to build the app step by step. Figure 1 below shows the building blocks of the application pipeline.



Figure 1. Head Pose Estimation Pipeline

The code for this project is available as a GitHub repository. You will first log in to the board, then download the repository to the board, **complete the script as per instructions** and finally run the experiments for the input image step-by-step.

1. Download the repository:

<https://github.com/Ascend-Huawei/experiment/tree/main/head_pose>

1. Models:

Pre-trained models are provided for 1. face detection 2. head pose estimation. Please follow the instructions below to convert the models to ‘Offline Model’ for inference on the board.

1. Download the pre-trained Face Detection (Caffe) model network and weights from the following links to the project directory 'head\_pose’.

**Weights**: <https://c7xcode.obs.cn-north-4.myhuaweicloud.com/models/face_detection/face_detection.caffemodel>

**Network**: <https://c7xcode.obs.cn-north-4.myhuaweicloud.com/models/face_detection/face_detection.prototxt>

Then, execute the following command from the same project directory ‘head\_pose’ to convert the model:

*atc --framework=0 --model="face\_detection.prototxt" --input\_shape="data:1,3,300,300" --weight="face\_detection.caffemodel" --input\_format=NCHW --output="face\_detection" --output\_type=FP32 --soc\_version=Ascend310*

1. Download the pre-trained Head Pose Estimation (Caffe) model network and weights to your project directory 'head\_pose'.

**Weights**:<https://obs-model-ascend.obs.cn-east-2.myhuaweicloud.com/head_pose_estimation/head_pose_estimation.caffemodel>

**Network**: <https://github.com/Ascend-Huawei/models/blob/master/computer_vision/object_detect/head_pose_estimation/head_pose_estimation.prototxt>

**Note**: To download the Network file with ‘wget’, please use following command:

*wget https://raw.githubusercontent.com/Ascend-Huawei/models/master/computer\_vision/object\_detect/head\_pose\_estimation/head\_pose\_estimation.prototxt*

Then, execute the following command from your project directory 'head\_pose' to convert the pre-trained model for head pose estimation to offline model (.om) format:

*atc --framework=0 --model="head\_pose\_estimation.prototxt" --weight="head\_pose\_estimation.caffemodel" --input\_shape="data:1,3,224,224" --input\_format=NCHW --output="head\_pose\_estimation" --output\_type=FP32 --soc\_version=Ascend310*

**NOTE**: If the model conversion process fails, please download the offline model directly from the following link: [Google Drive](https://drive.google.com/file/d/1vKyuRg_NIDBx2C-KxM9Mf_dGUhRj7qf9/view?usp=sharing)

1. Complete the lines of code in the script*‘head\_pose\_estimation.py’.*

Fill in the missing lines of code, according to the instruction provided in comments. You will ne ed to complete each of the 5 steps, explained in the comments. Put your code where you find the comment ‘### Your code here … ###’

**Step 1:** Initialize ACL and ACL Runtime

**Step 2:** Load the model for face detection and head pose estimation

**Step 3:** Performing inference: call the *execute()* method for face detection inference.

**Step 4:** Performing inference: call the *execute()* method for head pose inference.

[**HINT** for step 1, 2, 3, 4: You may refer to the sample code in the image classification project, to learn how to use the required APIs in the code: <https://github.com/Atlas200dk/sample_image_classification_c73_python>]

**Step 5:** Based on the estimated (pitch, yaw, roll) angles, determine the viewing direction:

‘Straight Ahead’: pitch, yaw and roll angles lie in certain ranges

‘Up’ or ‘Down’: pitch angle is out of the range

‘Left’ or ‘Right’: yaw angle is out of the range

‘Swing Left’ or ‘Swing Right’: roll angle is out of the range

Your task in **Step 5** is to figure out the **values for the ranges** by testing on the provided images in 'head\_pose’ (face1.jpeg, head\_bend.jpeg, head\_tilt.jpeg, head\_upward.jpeg, test\_human.jpg) and complete the code blocks in function *‘head\_status\_get()’* by assigning the viewing direction (‘Left’, ‘Right’, ‘Swing Left’, ‘Swing Right’ or ‘OK’) based on the estimated angle values. The range and code for pitch angle is provided as an example in *‘head\_status\_get()’.* Please refer to it and complete for yaw and roll.

To run the experiment with sample image files included in the repo, and verify the outputs:

To run the application, open up a terminal and navigate to the project folder:

**cd head\_pose**

Run the python script:

**python3 head\_pose\_estimation.py**

You may replace the input image (path to input image file given in variable ‘*img\_file’*) to see what the outcome is for a variety of head poses.

Expected outputs printed to terminal (sample):

**Head angles: [9.7900390625, 1.219940185546875, -1.763916015625]**

**Pose: Viewing direction: Straight ahead**

**Appendix**

The inputs and outputs of the models are described below:

**Face Detection Model:**

* **Model Inputs:**
* Input Shape: [1,300,300, 3]
* Input Format: NCHW
* Input Type: BGR FLOAT32
* **Model Outputs:**

2 lists. Only the 2nd list is used.

* 1st list shape: [1, 8]
* 2nd list shape: [1, 100, 8]

For the second list: ‘100’ represents 100 bounding boxes. ‘0-8’ describe information of each box as below:

* 0 position: not used
* 1 position: label
* 2 position: confidence score
* 3 position: top left x coordinate
* 4 position: top left y coordinate
* 5 position: bottom right x coordinate
* 6 position: bottom right y coordinate
* 7 position: not used

**Head Pose Estimation Model:**

* **Model Inputs:**
* Input Shape: [1, 3, 224, 224]
* Input Format: NCHW
* Input Type: BGR FLOAT32
* **Model Outputs:**

List of numpy arrays:

* Array shapes: (1, 136, 1, 1), (1, 3, 1, 1)
* The first list is a set of 136 facial keypoints. The second list in the output containing the 3 values of **[pitch, yaw, roll]** angles predicted by the model, which are used to determine head pose based on some preset rules.