Head Pose Tracker

INTRODUCTION:

Head pose estimation is a challenging problem because of the various steps required to solve it. Head pose is an important cue in computer vision when using facial information. Firstly, we need to locate the face in the frame and then the various facial landmarks. In this project, real-time human head pose estimation using TensorFlow and OpenCV has been implemented which can detect the direction person looking towards.

IMPLEMENTATION:

For the case of head pose estimation, it involves several steps. In this section, we briefly explain all the steps and their implementation.

Face detection:

The first step of the implementation was to detect all the faces present within the region of interest (ROI). For this project, it is assumed 80% of the image dimension as ROI. Any face which occupies less than 10% area of ROI has been discarded from being detected. The face that is close to the center of ROI is considered as the face under focus and has been shown with a green bounding box while the rest is shown in red.

Algorithms:

Haar-Cascade face detector:

This was the first algorithm that was tested for this project. It was easy to implement and works almost in real-time due to simple architecture and low computational requirements. But it was noticed that this algorithm was producing a lot of false predictions and it fails to predict non-frontal face (face with an angle). This was a major drawback for our project as detecting the non-frontal face was significant.

HoG Face Detection in Dlib:

This is widely used for simple face detection applications. For more information, one can read this. This had very low false prediction and was a fast algorithm. Though it did predict for slightly angled face, the prediction was not up to mark for a completely side face which is significant for head pose estimation.

Deep Neural Network face detector included in OpenCV:

OpenCV provides 2 models for face detection which was included in OpenCV from version 3.3. It is based on Single-Shot-Multibox detector and uses ResNet-10 Architecture as the backbone. For this project, we used the 8-bit quantized version using TensorFlow. For more information, one can read this. This did a good job for non-frontal faces and had solved the issue of false prediction.

Facial landmark detection:

The second step was to detect the facial landmarks on all the detected faces from the ROI. Landmarks will further help in estimating the pose of the head.

Algorithms:

Dlib facial landmark detection:

Dlib has a pre-trained model available for face landmark detection and its implementation can be found here. The model has been trained on the iBUG 300-W dataset. It predicts 68 landmarks on the face. Though this is the most used facial landmark detection it was not doing good for non-frontal faces. This was a major drawback.

CNN trained model for landmark detection:

This model had far better accuracy than the previous one for both frontal and non-frontal faces. This pre-trained model was found here. It also has been trained on the iBUG 300-W dataset and predicts 68 landmarks. This model was used in this project.

Head Pose Estimation:

The pose of an object refers to its relative orientation and position with respect to a camera. With the detected faces and facial landmarks, we can estimate the head pose.

For this project, the nose tip, chin, extreme left and right points of lips, and the left corner of the left eye and right corner of the right eye are the six points of the face that are considered. We estimate the rotational and translational vectors at the nose tip using standard 3D coordinates of these facial landmarks. 2D points are obtained from the facial landmark detection algorithm and 3D points for the same have been roughly estimated. This article gives a complete overview of this. We can project those 3D points onto a 2D surface, which is our image after we have the needed vectors.

With this, we can further calculate the theta and phi angle with respect to the nose tip. Once these angles are obtained, we can manually set the threshold to estimate the direction in which a person is looking.

RESULTS:

The implementation can detect all the faces that occupy more than 10% area of ROI. Further, the face that is close to the center is shown with the green bounding box while the rest is in red. The facial landmarks of all the faces are also displayed. Further, the implemented pose estimation algorithm calculates the theta and Phi angles using which it displays the direction in which the person is looking.

If the head goes out of focus and comes back in the algorithm will detect this and will track that face as the one in focus again as long as no other face is closer to the center of ROI. The algorithm does work with good accuracy even with foreign objects like headphones and spectacles as long as it doesn't majorly occlude the face. For slight occlusions, the algorithm performs pretty well. But with the major occlusions covering most of the face features the algorithm fails.