Assignment

Project 1 - Hand Detection for Safety in Machine Operation

System Architecture – The components that can be used to make a system to implement hand detection near critical machine parts are as follows:

- Camera – This becomes the main sensing unit for the system and captures the video of machine parts and its surroundings.
- NVIDIA Jetson Nano – After sensing, we need the brain of the system. The NVIDIA Jetson Nano is a small, powerful computer used in AI and embedded applications. This acts as the receiver of the video, processes it and performs the computer vision application of detecting as well as tracking hands in the frame.
- Output Sensors – Once the functionality is performed by Jetson Nano, there can be two outputs. When the hands are not in the frame, or they are in the frame but the distance between hands and the machine (or the critical part) is greater than some threshold, the output is safe, and sensors do not raise any flags.

  On the other hand, if the hands are in the frame and the distance between the hands and critical machine parts is less than the threshold, then the output will be danger, and sensors will raise the flag.

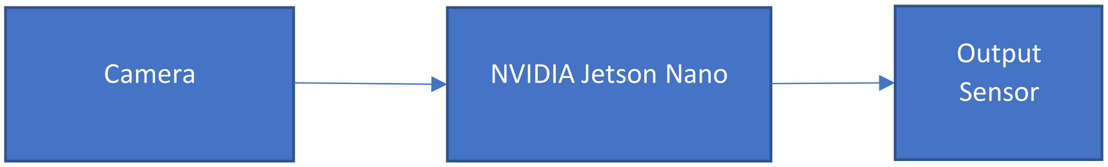

Image Processing and hand detection – I have used the in-built camera of my laptop as the primary sensor to capture videos for hand detection and tracking in real time.

I have used OpenCV and MediaPipe as my main libraries for image processing and hand tracking.

I have created a class called hand_detector in my code. This class initializes the hand detector from mediapipe library. The detection and tracking confidence I have used is 50%, this can be changed to increase the efficiency of the system. Currently, the model detects only 1 hand in the frame, this can be changed using the argument “maxHands” in the initialization function.

Firstly, we change the BGR image read by OpenCV to RGB image using cv2.cvtColor(). The results variable then stores the prediction made by the process function. Now, we access the hand landmarks using the results variable and draw the key points of the hand. MediaPipe considers 21 points for hand detection. Since we only consider 1 hand in this case, we obtain the key points of the hand number 0. This gives a list of coordinates along with ids of the key points. We draw a circle around these key points and hence hands are detected and tracked in the video.

Due to hardware constraint, I have considered a rectangular box or a vertical line to be a machine (or critical part). The idea is that, once the hands are detected using the above method, we continuously check the coordinates of key points and calculate their distance with the coordinates of the assumed machine/ machine part I.e., the vertical line. If the distance between key points and the line is greater than some threshold, there is no danger, and the operation can run normally. If the distance is less than some threshold, then we display a danger text on the video itself. In the actual operation, we can buzz an alarm or switch on the light over that particular machine.

ROI in this image can be machine part as well as hands. Since machine parts will be static, the coordinates won’t change and using those will not be a problem.

There is another method which we can use to detect and track hands which is by using deep learning. In this method we can use TensorFlow's object detection API for detecting hands with YOLO model. We can also estimate depth for comparing the relative positions of hands with respect to machines. This can help in preventing accidents. For the dataset we can use open datasets from Kaggle.

Project 2 - Process Monitoring Using Video Analytics

The system architecture of this project is similar to the above project except that this project also includes multiple sensors related to different processes so that we can keep track of key points related to each step.

The idea is to keep track of key points for each step. Once a step is complete, we make a virtual target box (coordinates are known) and compare it with the actual box formed by the coordinates of the target after that step. We apply IoU and if it is greater than some threshold then we move on to the next step otherwise we raise a flag. We also keep in mind the time to complete a step.