PROBLEM:

Identify shaking hands and print violation and save the image.

Abstract

The ability to track the human hand in real-time from monocular images has important implications for the design of unobtrusive, marker-free human-computer interaction paradigms. However, during real world interactions, the human hand is presented in multiple orientations, may be partially occluded from camera view, exbitit random motion and have large variations in skin tone making tracking difficult. Furthermore existing approaches to these challenges are limited (use of multiple sensors, dependence on manually engineered detection models) in terms of portability, ease of integration and scale. In this work, a HandTrack, an end-to-end trainable convolutional neural network model that learns to track the human hand using a single RGB image in realtime. I provide experimental results and code that demonstrates the qualitative and quantitative value of the model on a publicly available dataset. Finally, I provide an easy way to use library (Python) which application developers can import . After hand Detection I used intersection method to detect handshaking like if one hand object intersects other then prints violation and saves the image.

Motivation — Why Track/Detect hands with Neural Networks?

There are several existing approaches to tracking hands in the computer vision domain. Incidentally, many of these approaches are rule based (e.g. extracting background based on texture and boundary features, distinguishing between hands and background using color histograms and HOG classifiers, etc) making them not very robust. For example, these algorithms might get confused if the background is unusual or where sharp changes in lighting conditions cause sharp changes in skin color or the tracked object becomes occluded.

With sufficiently large datasets, neural networks provide opportunity to train models that perform well and address challenges of existing object tracking/detection algorithms — varied/poor lighting, diverse viewpoints and even occlusion. The main drawbacks to usage for real-time tracking/detection is that they can be complex, are relatively slow compared to tracking-only algorithms and it can be quite expensive to assemble a good dataset. But things are changing with advances in fast neural networks.

Furthermore, this entire area of work has been made more approachable by deep learning frameworks (such as the tensorflow object detection api) that simplify the process of training a model for custom object detection. More importantly, the advent of fast neural network models like ssd, faster r-cnn, rfcn etc make neural networks an attractive candidate for real-time detection (and tracking) applications. There are multiple applications for robust hand tracking like this across HCI areas (as an input device etc.).

Model

I have taken a trained model on online ‘hand_inference_graph’ which is trained using Egohands dataset.

The Egohands Dataset

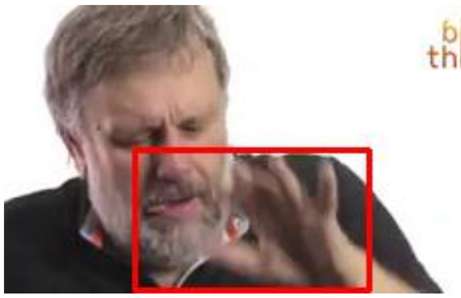
The hand detector model is built using data from the Egohands Dataset . This dataset works well for several reasons. It contains high quality, pixel level annotations (>15000 ground truth labels) where hands are located across 4800 images. All images are captured from an egocentric view (Google glass) across 48 different environments (indoor, outdoor) and activities (playing cards, chess, jenga, solving puzzles etc). The Egohands dataset (zip file with labelled data) contains 48 folders of locations where video data was collected (100 images per folder).

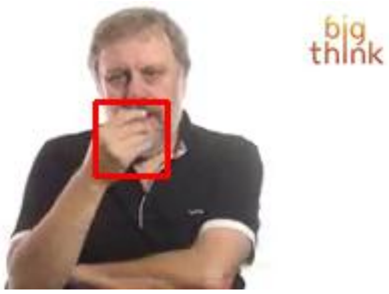
Using the Detector to Detect/Track hands

The general steps to detect hands are as follows:

- Load the `frozen_inference_graph.pb` trained on the hands dataset as well as the corresponding label map.
- Read in your input image (this may be captured from a live video stream, a video file or an image).
- Detect hands and visualize detected bounding detection_boxes.

Output:





Handshaking detection:

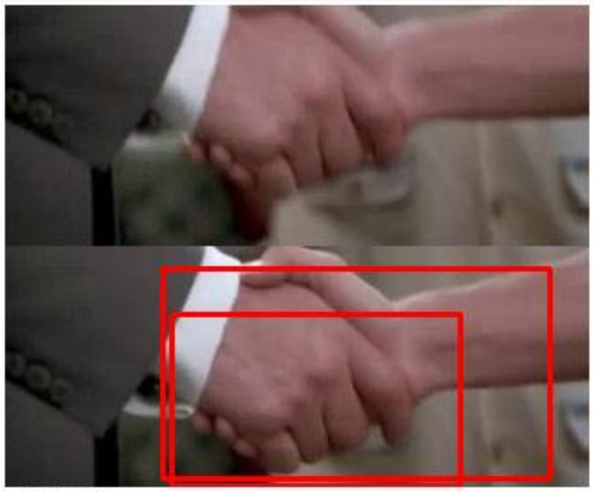
After detecting the hand objects, added objections to image. Created object touch function where it converts objects to polygons and one polygon is compared to other using intersects function .If intersects ,violation is printed or a beep sound and the frame is saved. Like this each hand object is compared to all other hand object.

Output:

The obtained output is not accurate .There is small bug in printing violation. One object checks the intersection for same object, so it returns true when hand is detected and prints violation. When different object intersects it prints violation two time (like shown below). I tried to fix this error by checking only different objects(!=) but this didn't work .I don't know why .

```python
    # When everything done, release the capture
    cap.release()
    cv2.destroyAllWindows()

if __name__ == '__main__':
    con_detect()
```



```
violation
violation
```