Hands Tracker and Classifier

as a solution for detecting autistic behavior

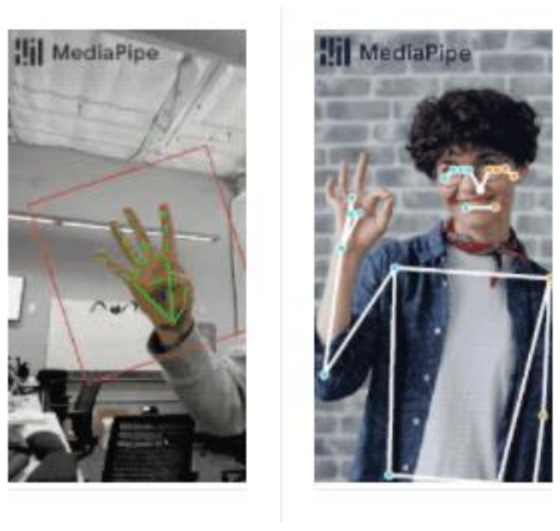
Guided by:

Professor **Hagit Hel-Or**

Students:

Julian Mour

Fadi Khateeb



University of Haifa – Etgar

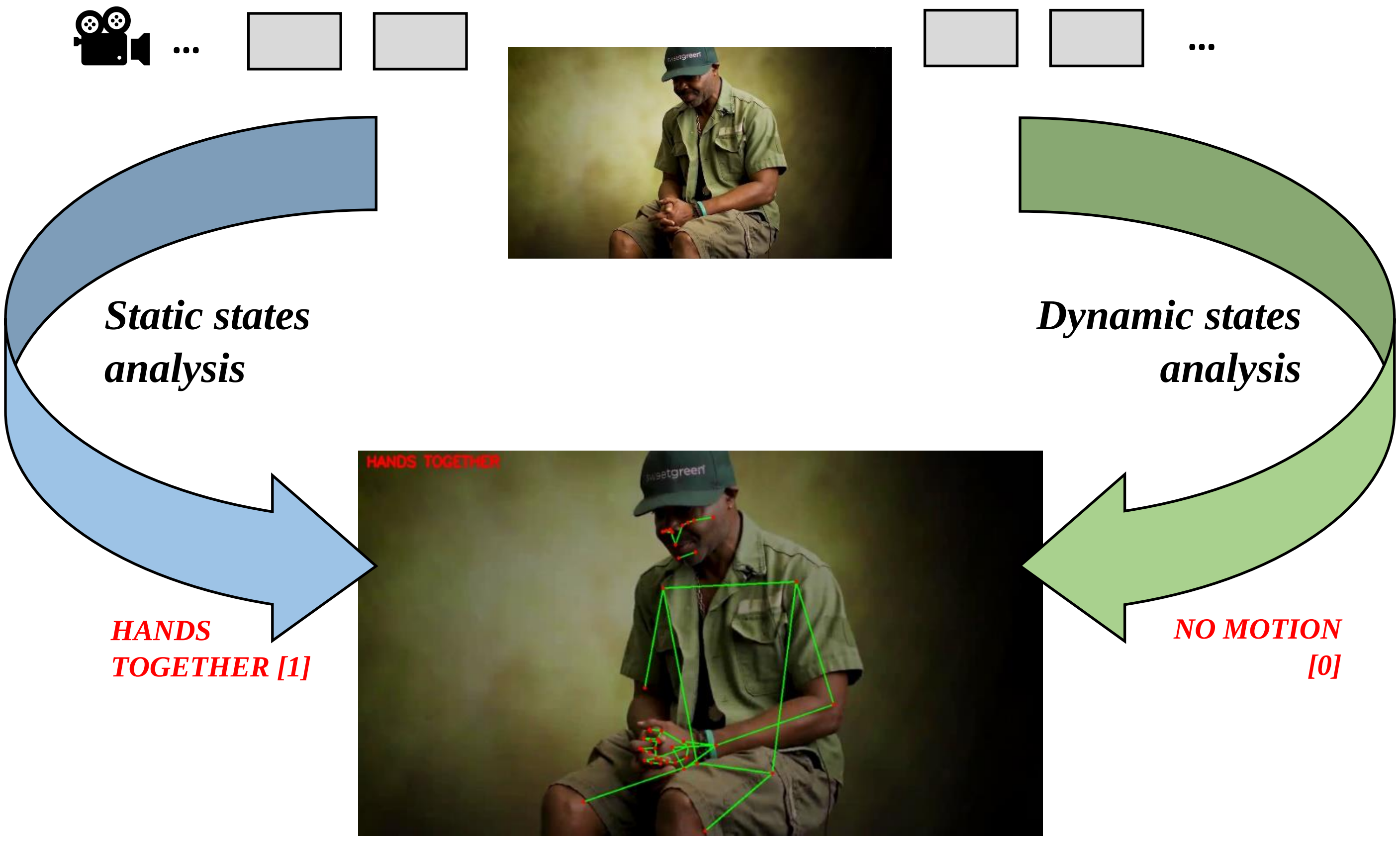
2020-2021

Problem stating

One of the significant signs that can point to an autistic behavior in a certain person is their body language, specifically their hands movements.

Tracking a patient's hands behavior by eye isn't an easy task to do, as there are a lot of states were the hands behavior needs to be recorded, whether it's in a static state (hands on legs, hands together…) or in a dynamic state (motion).

Using *Computer Vision* and *Machine Learning*, we can track a patient's hands behavior from a video and classify it to different states during the whole video.



Project Definition

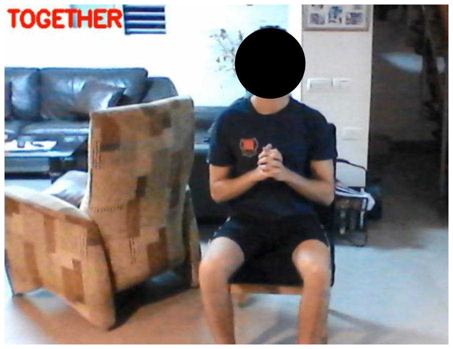
We used in our project two trackers we came across from an open-source code from ***Mediapipe***, which offers cross-platform, customizable ML solutions for live and streaming media. Mediapipe was developed by ***Google***.

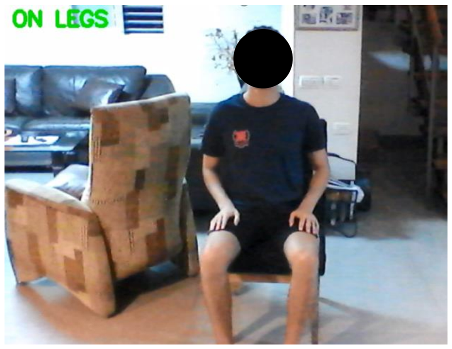
We used Mediapipe-Hands for hands detecting and tracking and Mediapipe-Pose for the body's pose detecting and tracking.

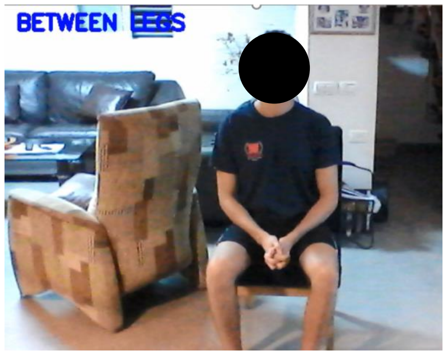
Our first mission was to be able to classify between two simple hand states. As mentioned before, *Machine Learning* was involved in our project; we developed a classifier that could classify a detected hand from a single frame to *open* or *closed*.

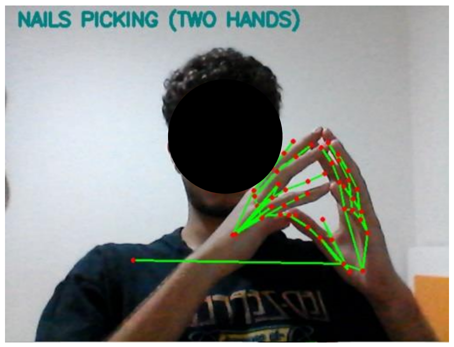
Our second mission was to be able to detect and classify static states of the hands for each frame. Using the hands and pose trackers from Mediapipe and the open-closed classifier we developed earlier, we were able to classify a frame from the input video to these static classes:

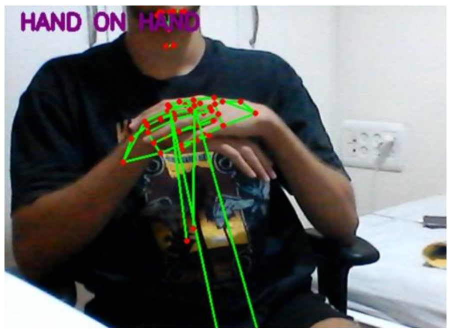
1. HANDS TOGETHER
2. HANDS ON LEGS
3. HANDS BETWEEN LEGS
4. NAILS PICKING (TWO HANDS)
5. HAND ON HAND
6. NAILS PICKING (ONE HAND)

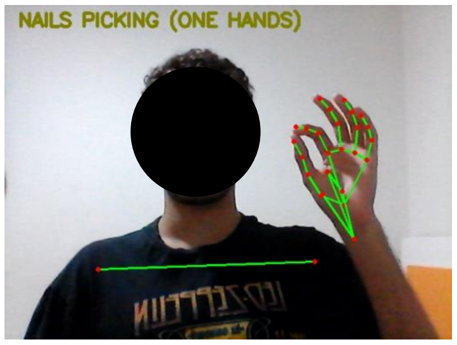












If no state from the above was detected, then the class chosen for this frame is:

0. <NO STATIC CLASS DETECTED>

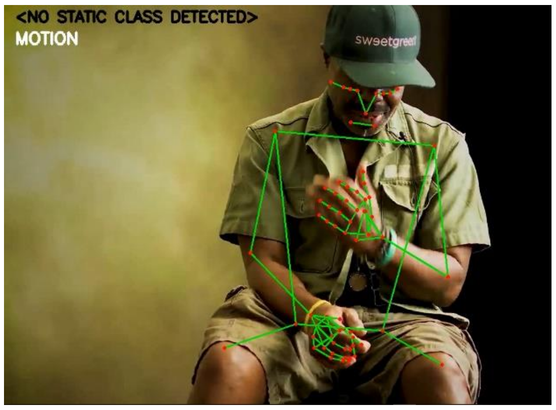
It's important to mention that in case of a motion of at least one hand in a certain frame, classes 1-5 cannot be given to this frame. In case of a motion of both hands, class 6 also cannot be given to this frame and the chosen class is 0.

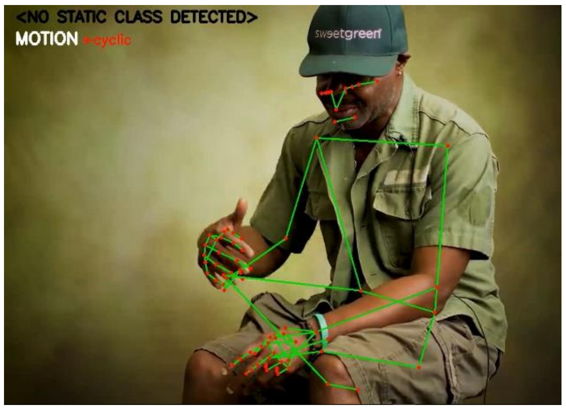
Our second mission was to be able to detect and classify dynamic states of the hands for each frame. There are only two possible states:

1. MOTION
2. CYCLIC MOTION (repetitive motion)

Or if there is no motion; class assigned is 0 for the certain frame.

Class 1 is only assigned if there is motion that doesn't have a cyclic pattern. Class 2 is assigned when there is motion and a cyclic pattern of it was detected.



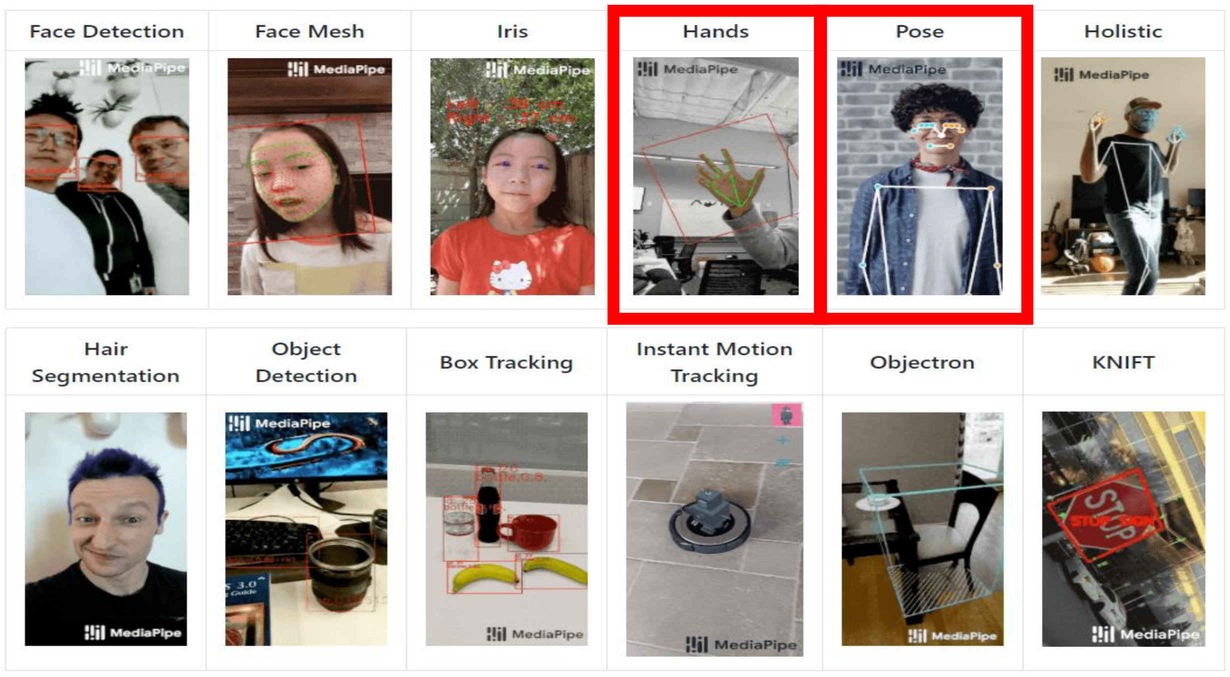


To conclude: The user passes the path of an input video to the code or run it in real time. The user also has the option to choose only one side of the video to be tracked: left or right (more about user’s instructions are at the end of the report and in file README.txt attached with code). Each frame of the input video will be classified to a static class (0-6) and a dynamic class (0-2). These results will be written into an *Excel file* at the end of the runtime. In addition, an output video will be built with the classes displayed for each frame, and the hands and pose trackers landmarks drawn on each frame.

Implementation

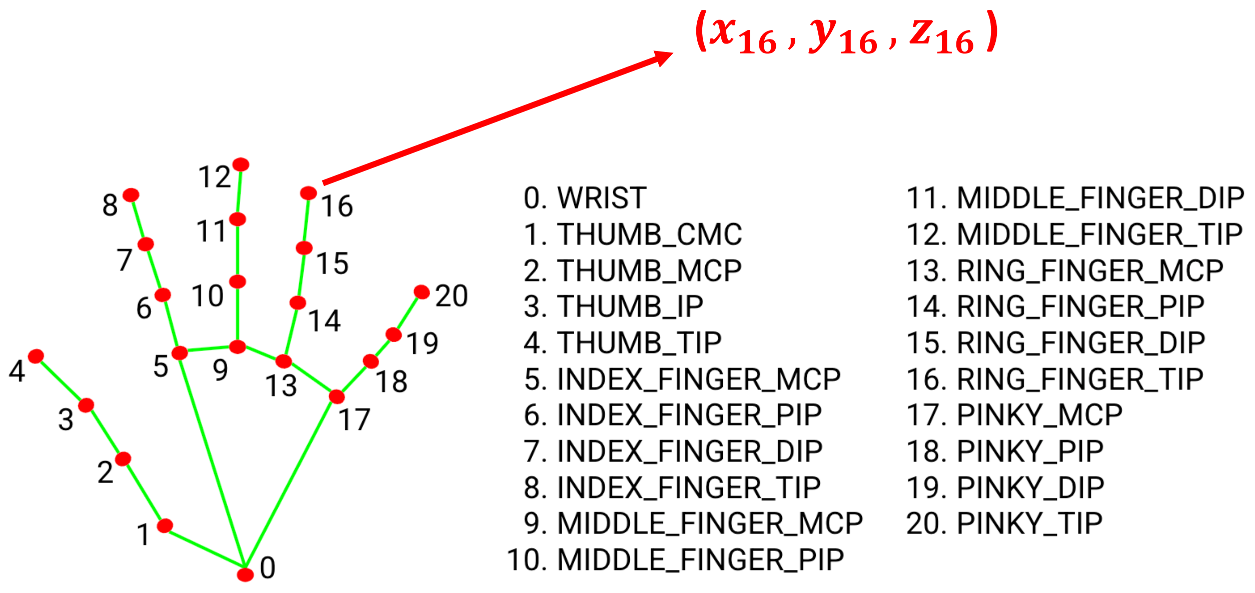
About Mediapipe trackers:

- *MediaPipe* offers cross-platform, customizable ML solutions for live and streaming media.
- Was developed by Google



1. *Mediapipe Hands:*

MediaPipe Hands is a high-fidelity hand and finger tracking solution. It employs machine learning (ML) to infer 21 3D landmarks of a hand from just a single frame.

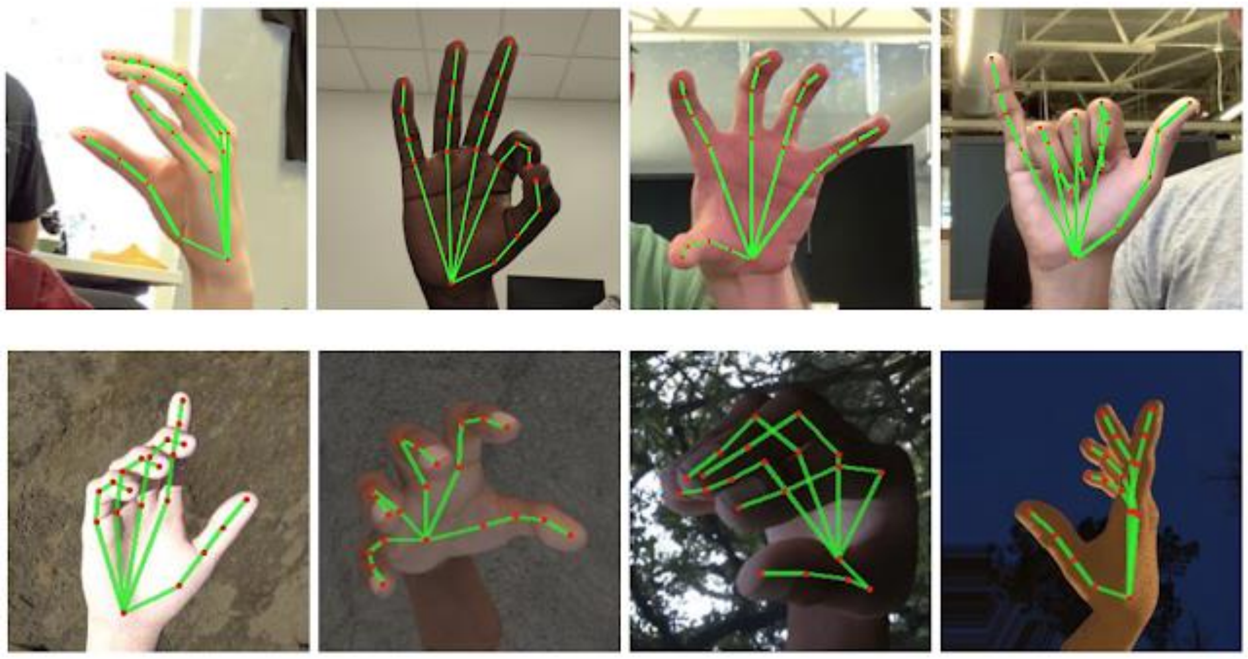


0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP

More information about the tracker can be found here:

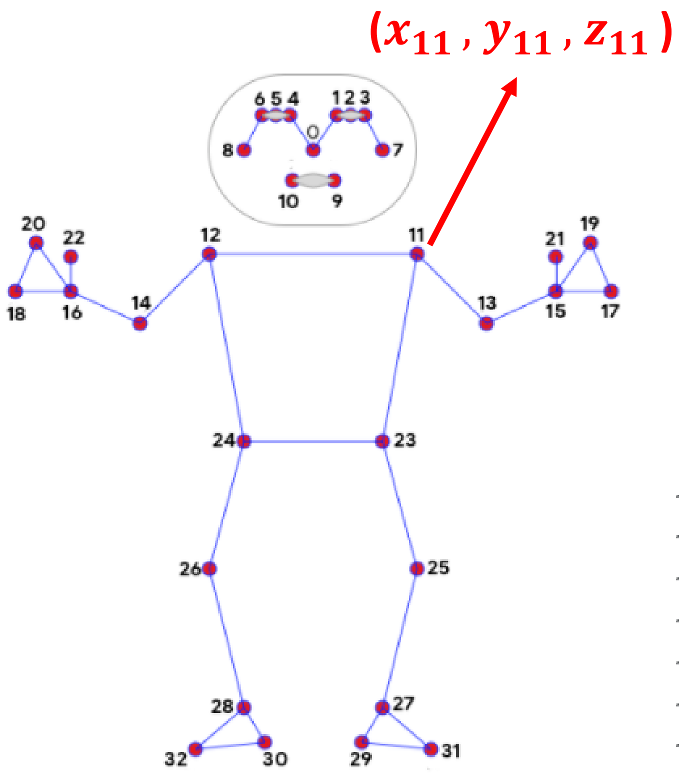
https://google.github.io/mediapipe/solutions/hands.html

Tracker output:



2. Mediapipe Pose:

MediaPipe Pose is a ML solution for high-fidelity body pose tracking, inferring 33 3D landmarks on the whole body from RGB video frames.

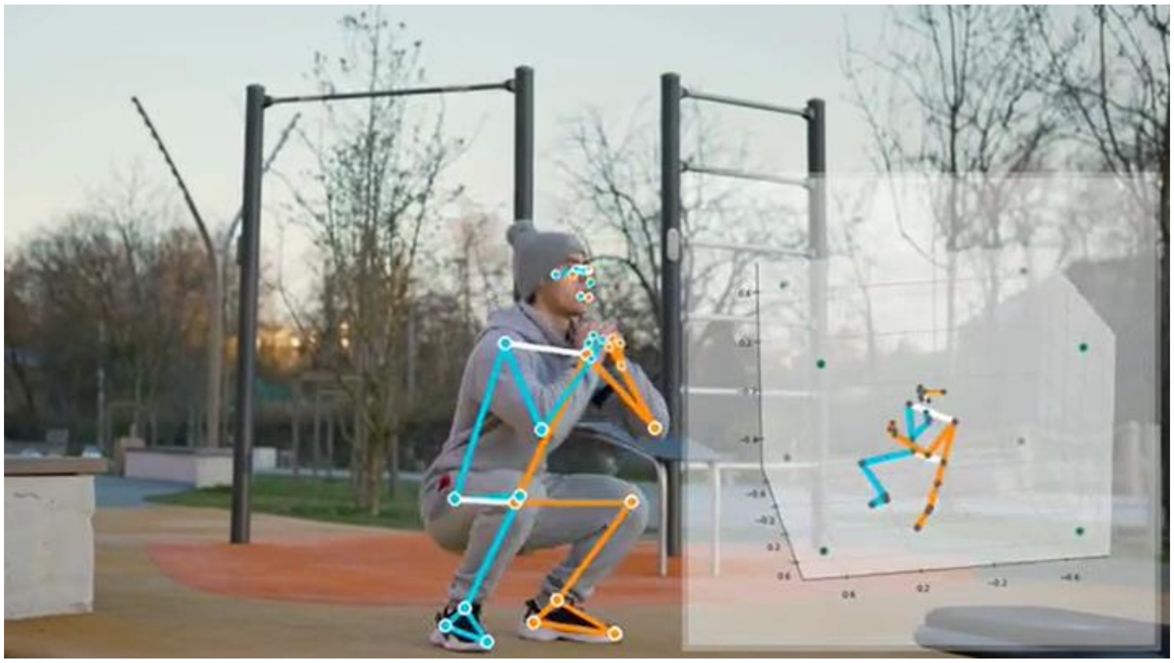


0. nose
1. left_eye_inner
2. left_eye
3. left_eye_outer
4. right_eye_inner
5. right_eye
6. right_eye_outer
7. left_ear
8. right_ear
9. mouth_left
10. mouth_right
11. left_shoulder
12. right_shoulder
13. left_elbow
14. right_elbow
15. left_wrist
16. right_wrist
17. left_pinky
18. right_pinky
19. left_index
20. right_index
21. left_thumb
22. right_thumb
23. left_hip
24. right_hip
25. left_knee
26. right_knee
27. left_ankle
28. right_ankle
29. left_heel
30. right_heel
31. left_foot_index
32. right_foot_index

More information about the tracker can be found here:

https://google.github.io/mediapipe/solutions/pose.html

Tracker output:



The open-closed classifier:

Our mission was to build a simple classifier that can classify a detected hand to *closed* or *open*. We filmed a couple of hands videos and applied the Medipipe hands tracker on them and used the output data of each frame (the landmarks of a detected hand) in our learning.

These are the learning details:

- **<u>The data:</u>** 21x21 matrix (***distances***) of the <u>normalized</u> distances between the landmarks of a single hand.

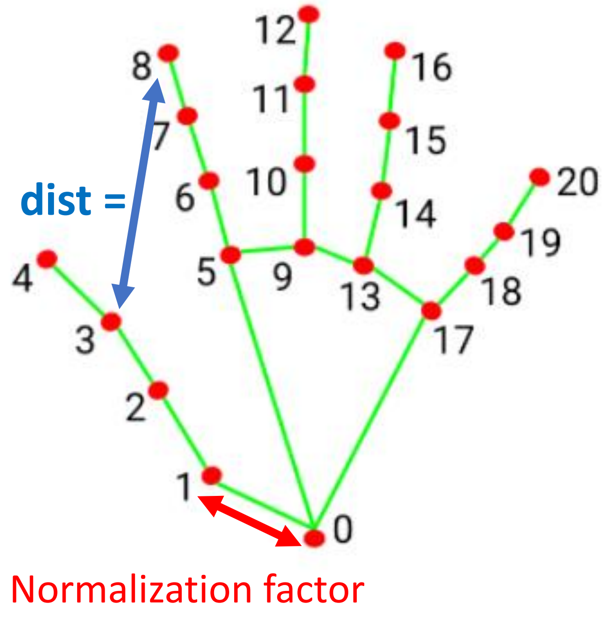
distances[3][8] =

distances[8][3] =

$$\frac{\mathit{dist}}{\mathit{normalization\ factor}}$$

distances[i][i] =

0



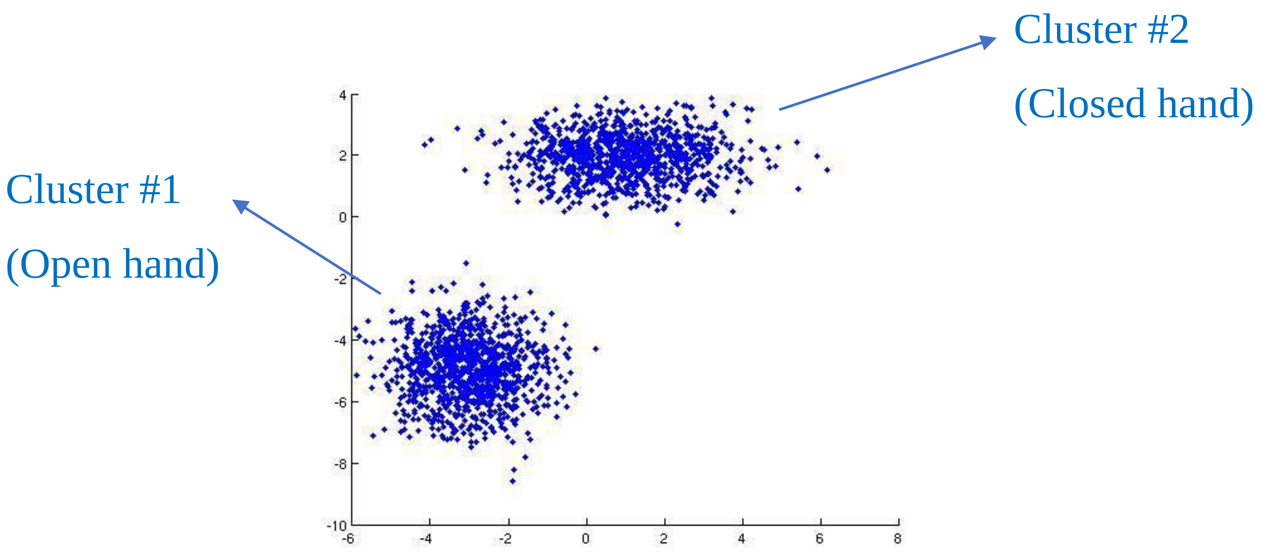
The normalization is done to avoid distance from camera differences. We chose the 0-1 normalization factor because the distance between the 0 and 1 landmark is nearly constant.

- **<u>Model:</u>** K-means with 2 clusters (unsupervised).

  About k-means in general:

  “*k*-means clustering is a method of vector quantization, originally from signal processing, that aims
  to partition *n* observations into *k* clusters in which each observation belongs to the cluster with the
  nearest mean (cluster centers or cluster centroid), serving as a prototype of the cluster.”

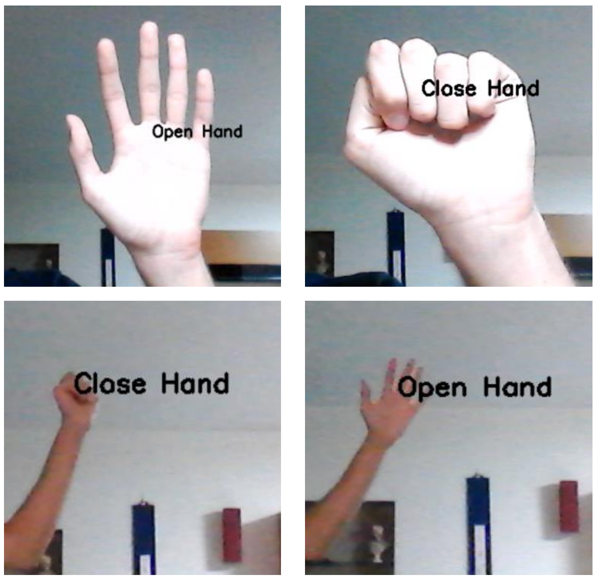
In our case, K=2:

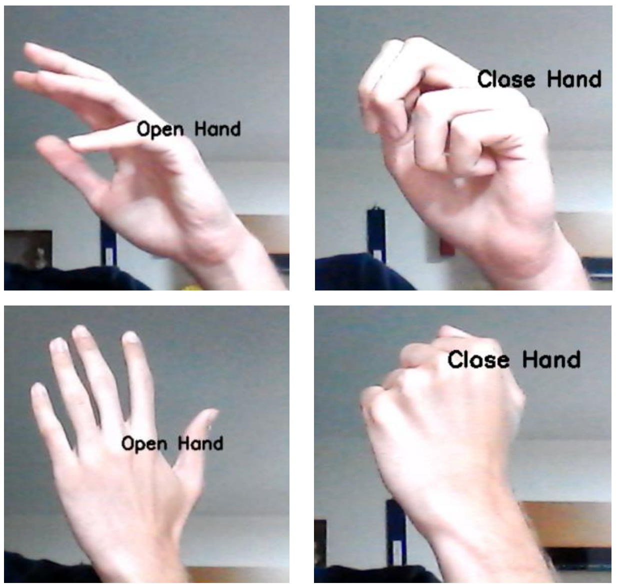


More information about K-means can be found here:

https://en.wikipedia.org/wiki/K-means_clustering

The classifier's results:





The static states detector:

Each batch of 30 frames is passed to a classifier that will give all 30 frames a certain static class from these classes:

0. <NO STATIC CLASS DETECTED>
1. HANDS TOGETHER
2. HANDS ON LEGS
3. HANDS BETWEEN LEGS
4. NAILS PICKING (TWO HANDS)
5. HAND ON HAND
6. NAILS PICKING (ONE HAND)

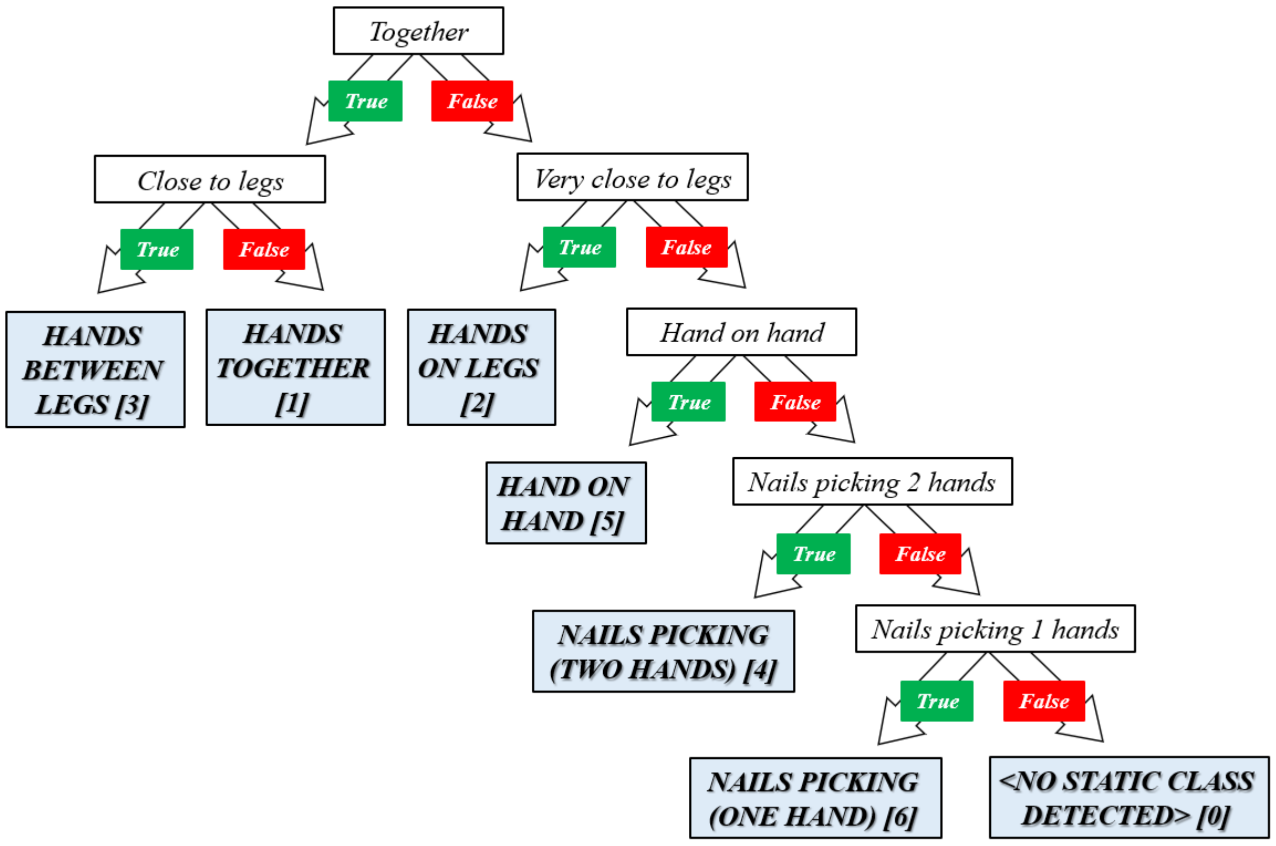
First, a couple of Boolean variables are calculated using the Mediapipe Hands tracker and the Mediapipe Pose tracker, as well as the open-closed classifier we developed earlier:

- *Together* = if the distance between the palms is beyond a certain threshold (0.4) and all hands are closed (an undetected hand is considered closed).
- *Close to legs* = if the right leg distance from the right hand and the left leg distance from the left hand are beyond a certain threshold (1.7).
- *Very close to legs* = if the right leg distance from the right hand and the left leg distance from the left hand are beyond a certain threshold (1.3).
- *Nails picking 2 hands* = if the distance of each finger of the left-hand from the corresponding finger of the right-hand is beyond or equal to a certain threshold (0.6).
- *Nails picking 1 hand* = if the distance of the thumb from the index finger is beyond a threshold (0.9) and the hand is

open. This must apply for at least one of the detected hands.

- *Hand on hand* = if the average of the distances of the three palm landmarks on one hand from the corresponding landmarks on the other hand is beyond a certain average (0.1).

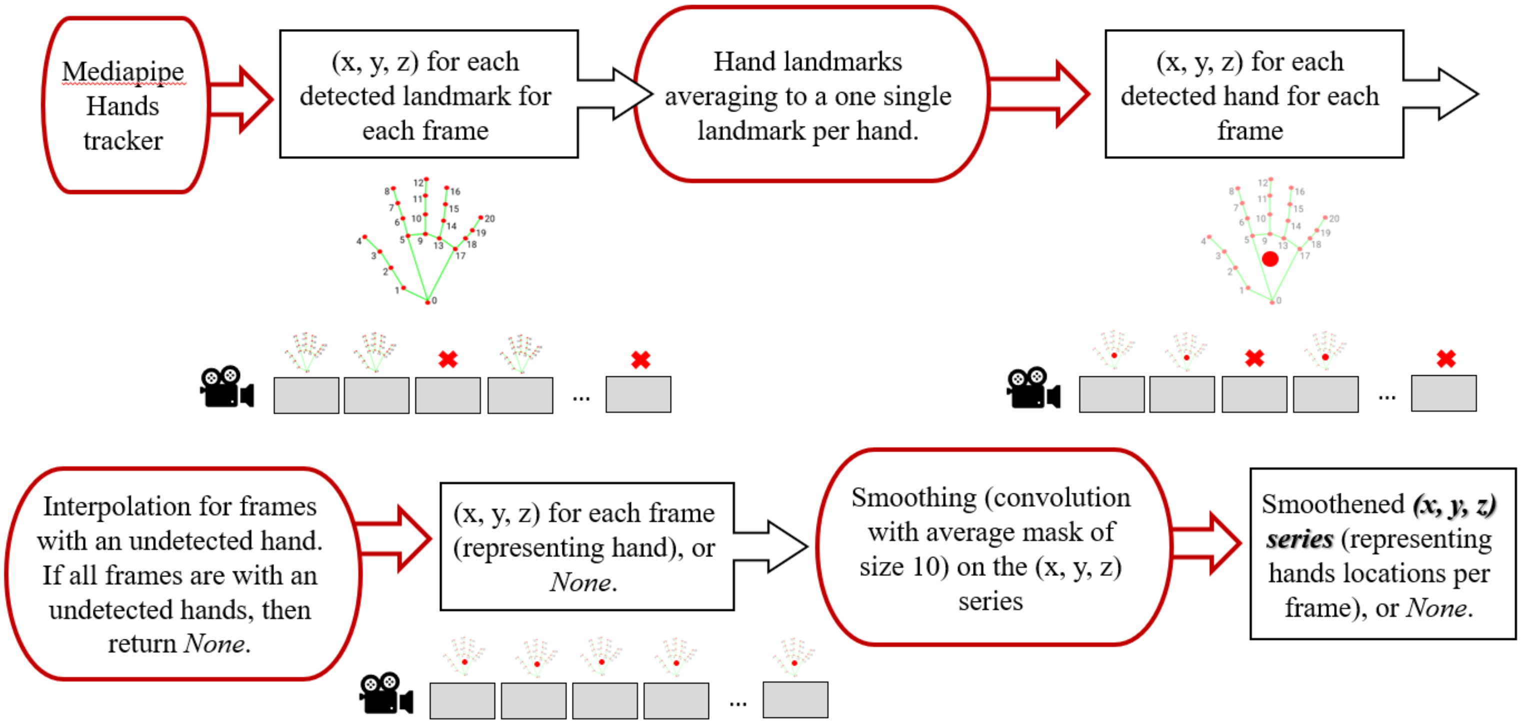
After calculating the variables above, we pass each frame from the batch of 30 frames to a decision tree that will give it a specific static class (0-6):



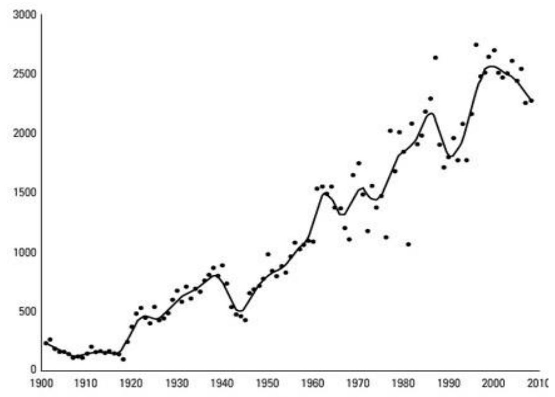
At last, given 30 classifications for the 30 frames, we check if there are enough frames (20% of the batch) that vote for a certain static class that isn't 0 (if there is more than one, then the one with the higher votes is chosen). If so, this class is given to the whole 30 frames batch. Otherwise, the class 0 (<NO STATIC CLASS DETECTED>) is given to the whole 30 frames batch.

The dynamic states detector:

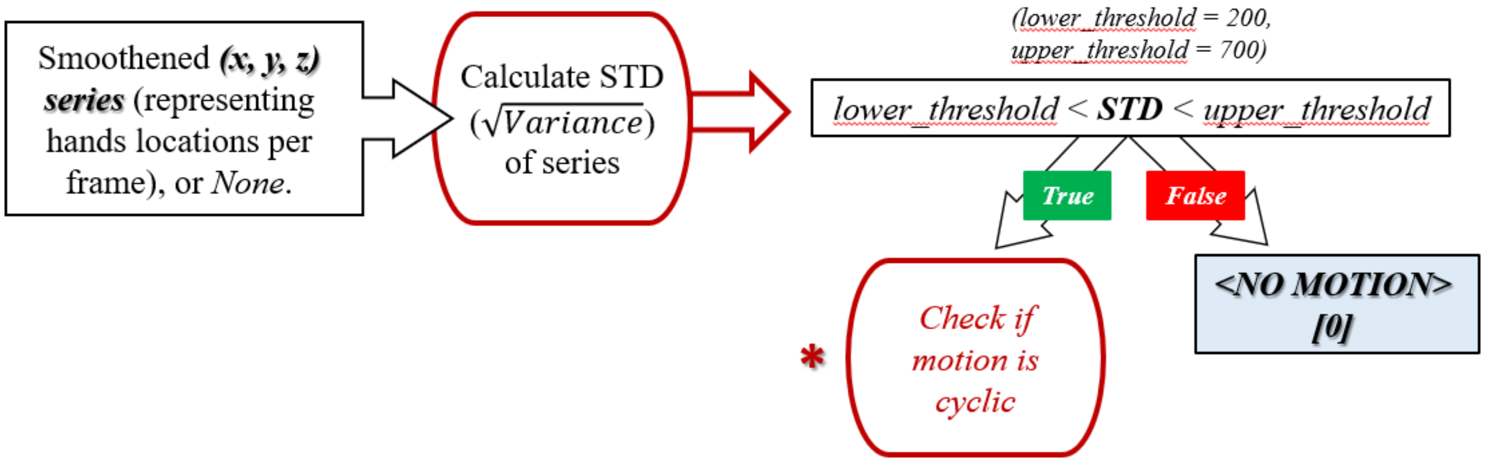
At first, given the data of the Mediapipe Hands tracker for each frame, we take each 90 frames batch and prepare it for dynamic classification:


Mediapipe Hands tracker
(x, y, z) for each detected landmark for each frame
Hand landmarks averaging to a one single landmark per hand.
(x, y, z) for each detected hand for each frame
Interpolation for frames with an undetected hand. If all frames are with an undetected hands, then return *None*.
(x, y, z) for each frame (representing hand), or *None*.
Smoothing (convolution with average mask of size 10) on the (x, y, z) series
Smoothened ***(x, y, z) series*** (representing hands locations per frame), or *None*.

The smoothing part is done to avoid tracking errors.

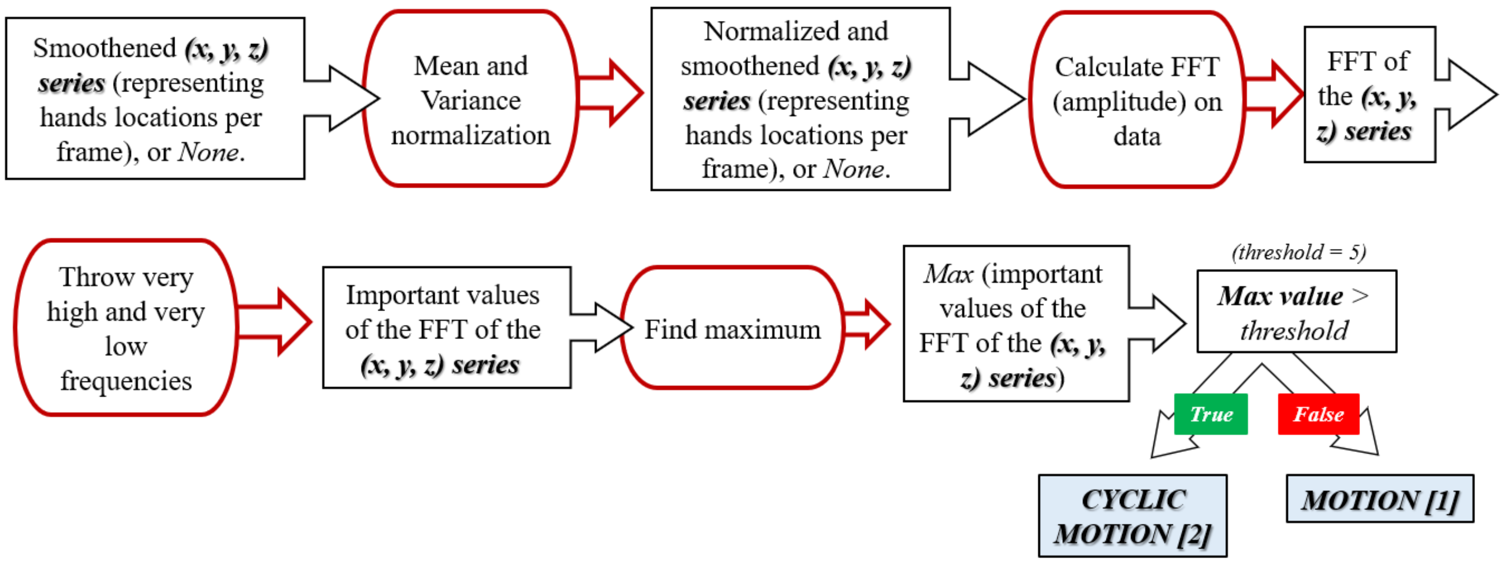


After getting the smoothened (x,y,z) series, we need to check if there is any motion in this series of frames:


Smoothened *(x, y, z)* ***series*** (representing hands locations per frame), or *None*.
Calculate STD ($\sqrt{Variance}$) of series
(lower_threshold = 200, upper_threshold = 700)
lower_threshold < **STD** < upper_threshold
True
False
Check if motion is cyclic
*
<NO MOTION> [0]

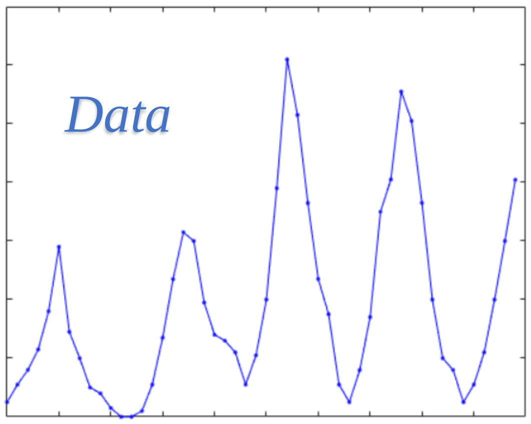
Variance is a measure of dispersion, meaning it is a measure of how far a set of numbers is spread out from their average value. Therefore, a very low variance of our series will be given when all landmarks in the different frames are around the same spot, when there is no significant motion. A very high variance will be given when all landmarks in the different frames are very far from each other. This will happen when the Mediapipe Hands tracker fail to detect the real hand in the frame but detect a false hand (on a background object for an instance).

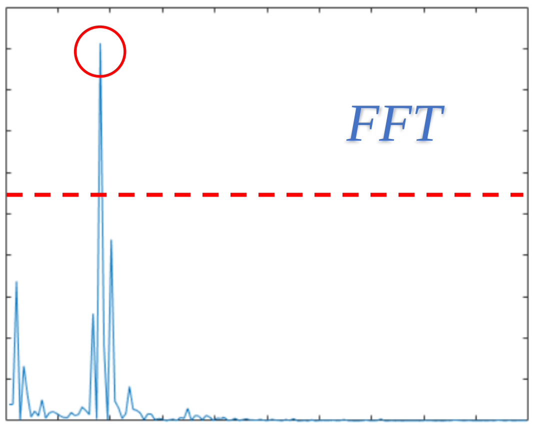
If our variance is somewhere in between, then we can say that there is a significant motion in these 90 frames. We now need to check if this motion is cyclic:


Smoothed (x, y, z) series (representing hands locations per frame), or None.
Mean and Variance normalization
Normalized and smoothened (x, y, z) series (representing hands locations per frame), or None.
Calculate FFT (amplitude) on data
FFT of the (x, y, z) series
Throw very high and very low frequencies
Important values of the FFT of the (x, y, z) series
Find maximum
Max (important values of the FFT of the (x, y, z) series)
(threshold = 5)
Max value > threshold
True
False
CYCLIC MOTION [2]
MOTION [1]

Basically, we are checking if the Fourier values contain a significant peak. If so, this means that the data (the hand route) is most likely repetitive along most frames, which could indicate a cyclic pattern in this peak frequency.

For more explanation and an example about analyzing cyclic data with FFT, you can check the link below:





https://www.mathworks.com/help/matlab/math/using-fft.html

Instructions

1) General instructions and pre-running instructions:

Please first follow general instructions to install MediaPipe Python package:

MediaPipe offers ready-to-use yet customizable Python solutions as a prebuilt Python package.

MediaPipe Python package is available on PyPI for *Linux*, *macOS* and *Windows*.

You can, for instance, activate a Python virtual environment:

```
$ python3 -m venv mp_env && source mp_env/bin/activate
```

Install MediaPipe Python package and start Python interpreter:

```
(mp_env)$ pip install mediapipe
(mp_env)$ python3
```

For programming uses:

In Python interpreter, import the package and start using one of the solutions:

```
>> import mediapipe as mp
>> hands = mp.solutions.hands
>> pose = mp.solutions.pose
...
```

Tip: Use command deactivate to later exit the Python virtual environment.

for more information about MediaPipe's general instructions and ready-to-use python solutions, visit:

https://google.github.io/mediapipe/getting_started/python.html

for more information about MediaPipe in general, visit:

https://google.github.io/mediapipe/

2) How to run:

Run line in command terminal:

```
$ python testing.py <SIDE> <INPUT_VIDEO>(optional) <OUTPUT_VIDEO>
```

*<**SIDE**>:*

-*1* = both sides tracking (whole frame size)

0 = left side tracking only

1 = right side tracking only

*<**INPUT_VIDEO**>*(optional): Name of input video (with extension).

Input video must be in the same directory as the code files.

If this argument isn't passed, then real time tracking will be applied.

*<**OUTPUT_VIDEO**>*: Name of output video (without extension).

Output video will be created in the same directory as the code files.

an example for a line to run:

```
$ python testing.py -1 interview2.mp4 interview2_out
```