

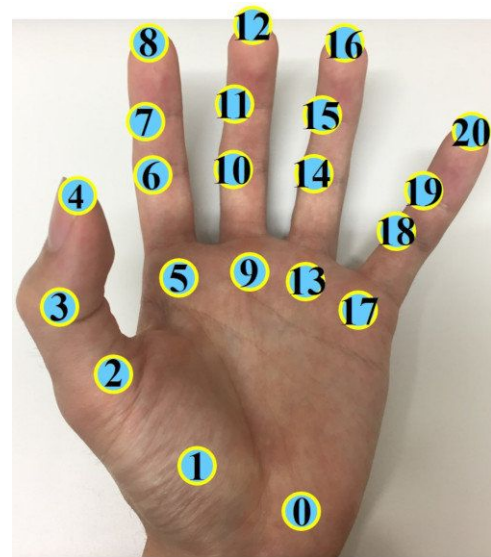
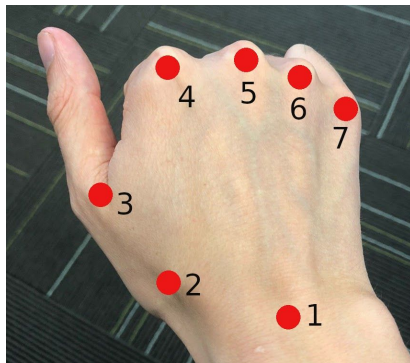
MediaPipe Hand Tracking Model

1. First detects palm in whole image using 7 keypoints.
2. Constructs a bounding box around the hand, rotated to align.
3. Detects 21 2D keypoints on cropped hand image.

[Article](#)

[Documentation](#)

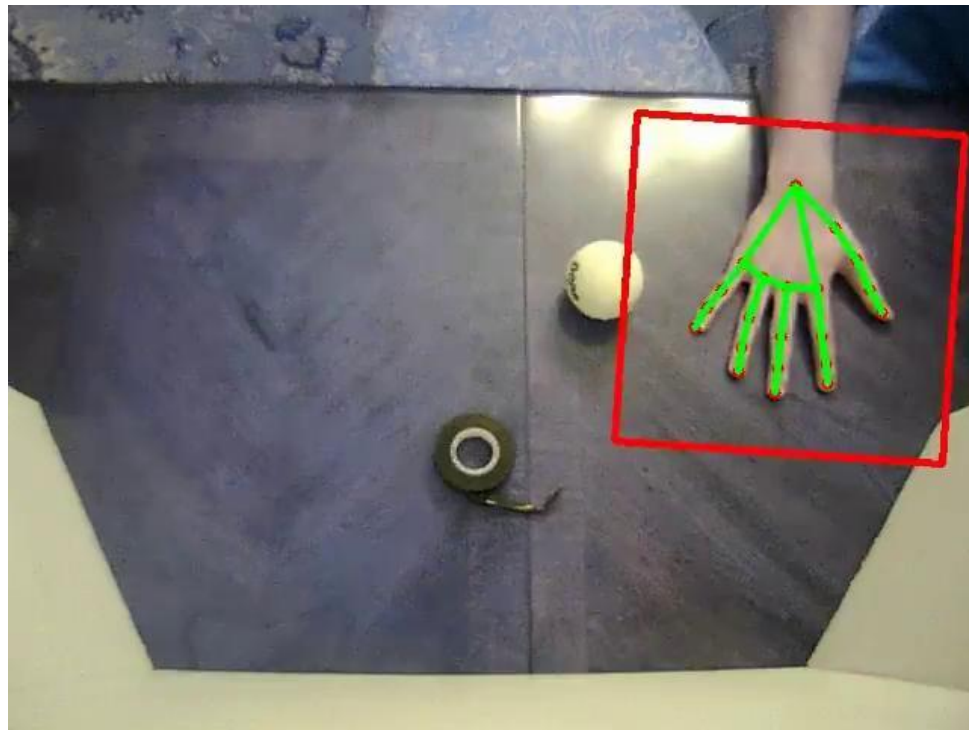
[Git](#)



Evaluating Model Performance

- 7 different types of grasping exercises were performed and recorded.
- Pose approximation was performed using the MediaPipe model and results were overlaid onto the input video.

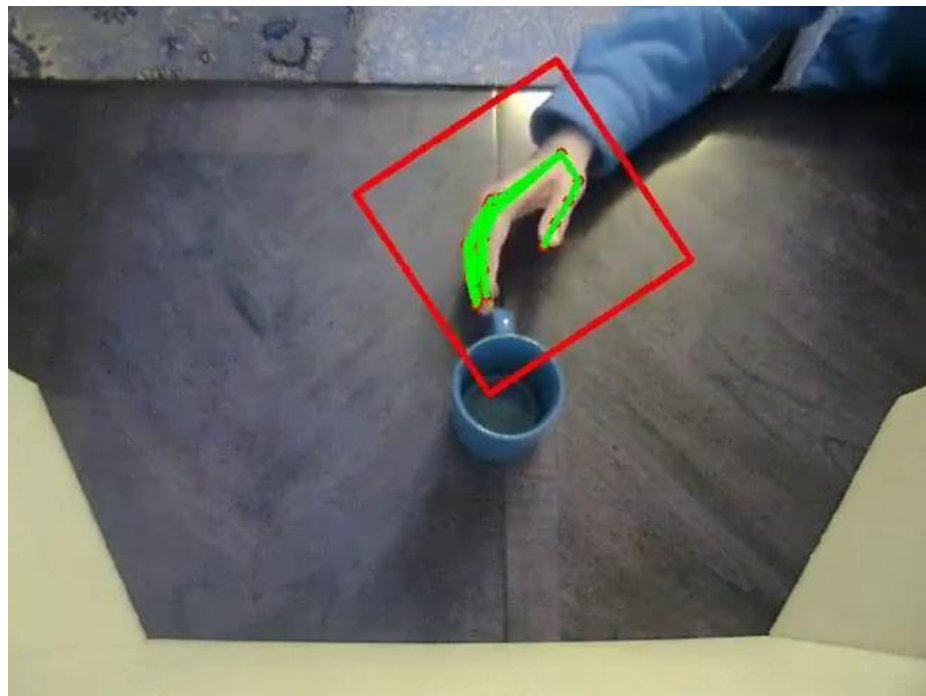
Circular Grasp



Cylindrical Grasp



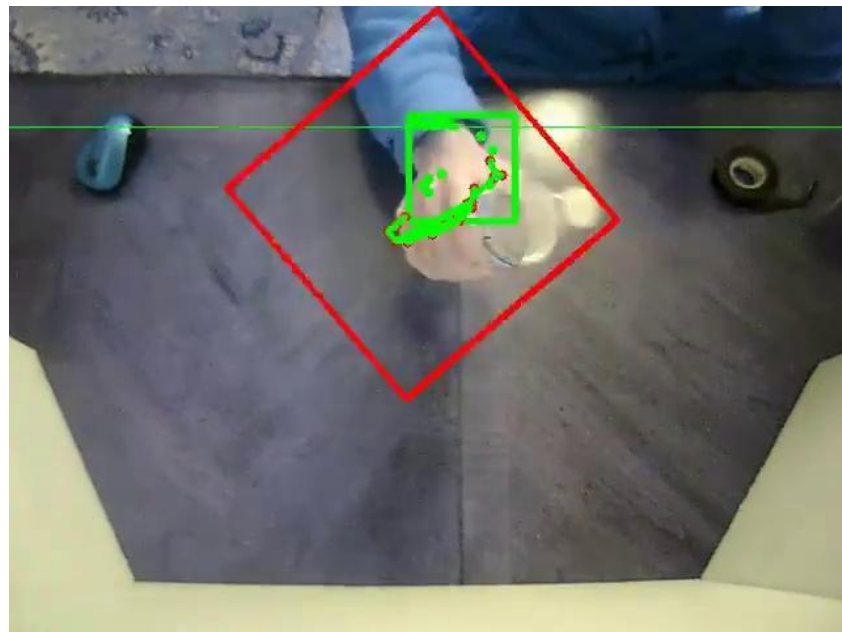
Hook Grasp



Tripod Grasp



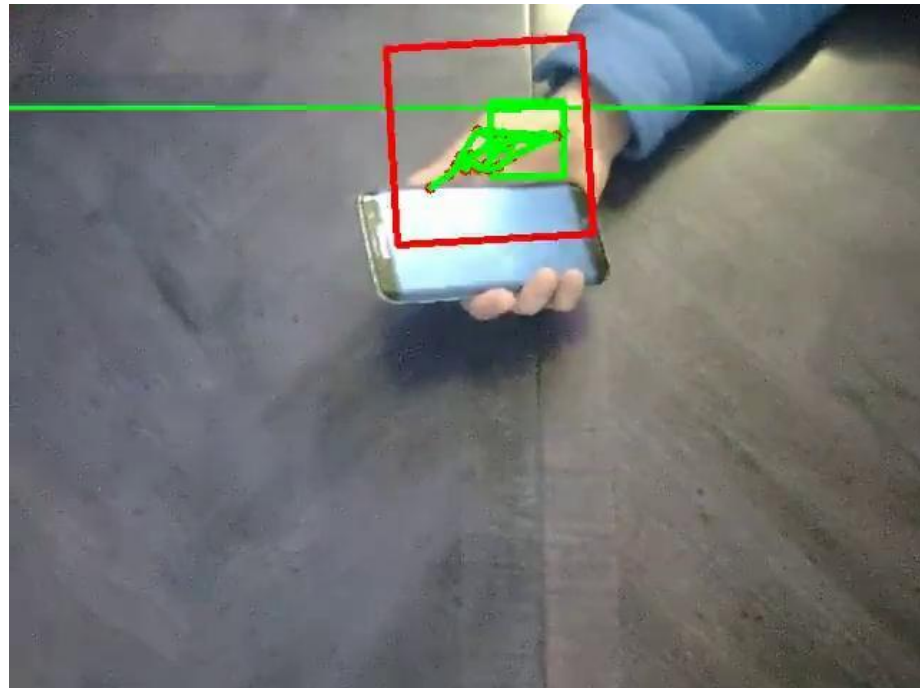
Bottle Cap



Doorknob

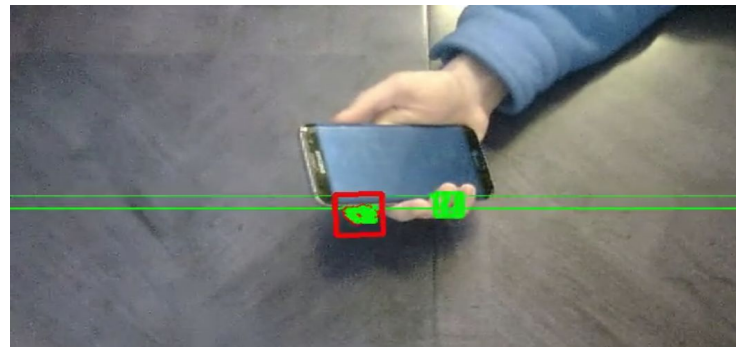
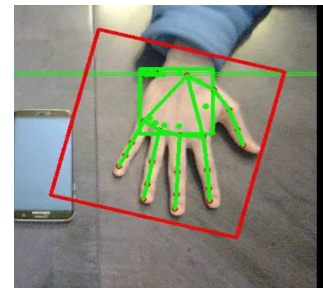


Power Grasp



Reliability

- 'Jitter' evident when hand is being tracked properly
 - Denoising or Savgol Filter
- When palm and the majority of fingers are obscured, tracking is lost.
- Model resumes tracking as soon as palm becomes visible again.



Tensorflow.js Version

- Raw data from the model

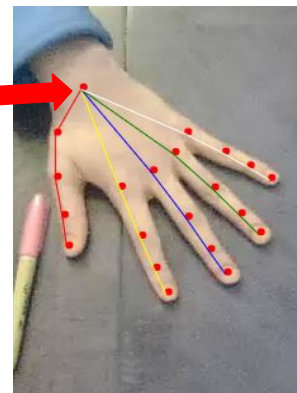
- annotations
- bounding box
- detection confidence
- 21 keypoints

```
▶ annotations: {thumb: Array(4), indexFinger: Array(4), middleFinger: Array(4),
▶ boundingBox: {topLeft: Array(2), bottomRight: Array(2)}
  handInViewConfidence: 0.9999984502792358
▼ landmarks: Array(21)
  ▶ 0: (3) [257.68052376847055, 348.052025858178, -0.0009048506617546082]
  ▶ 1: (3) [294.141561264282, 324.24861499985974, -21.623380661010742]
  ▶ 2: (3) [308.2624509808577, 281.8110588521106, -31.607397079467773]
  ▶ 3: (3) [286.29726053595476, 241.97447463336164, -39.511173248291016]
  ▶ 4: (3) [250.21845967590568, 227.9144969069157, -46.125614166259766]
  ▶ 5: (3) [288.9500263051589, 232.2462773554418, -2.7760396003723145]
  ▶ 6: (3) [289.00851194141137, 206.45496768073014, -21.863893508911133]
  ▶ 7: (3) [285.7332254251631, 218.13747008099813, -35.343284606933594]
  ▶ 8: (3) [277.2641268702127, 230.55045259863698, -39.65968704223633]
  ▶ 9: (3) [258.4126597996791, 233.67017032995147, 1.0947312116622925]
  ▶ 10: (3) [258.6928720883687, 219.39323684337268, -23.551101684570312]
  ▶ 11: (3) [267.4288586224417, 257.3061589490663, -31.6258602142334]
  ▶ 12: (3) [268.808955494885, 275.5022191641041, -25.11463737487793]
  ▶ 13: (3) [232.35317399875902, 243.23545845389506, 0.4451589584350586]
  ▶ 14: (3) [232.49539020449689, 239.46285815754612, -25.228757858276367]
  ▶ 15: (3) [246.61275718523763, 276.6157192279348, -28.19062042236328]
  ▶ 16: (3) [249.6478297718215, 285.9235642448217, -20.95989227294922]
  ▶ 17: (3) [209.52741340033793, 256.42767863562386, -1.4947261810302734]
  ▶ 18: (3) [212.04174024550747, 252.5138496336699, -22.11198616027832]
  ▶ 19: (3) [225.94060656727882, 279.9903864078307, -22.973800659179688]
  ▶ 20: (3) [228.01503410274623, 289.23740460481946, -16.91110610961914]
  length: 21
```

Annotations

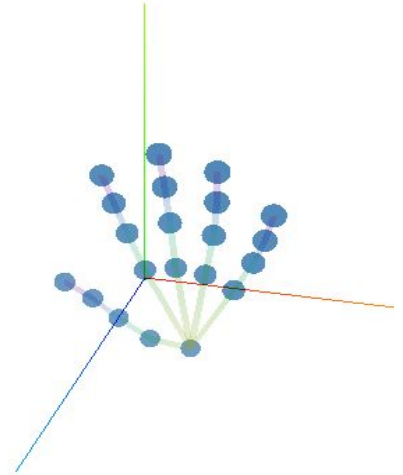
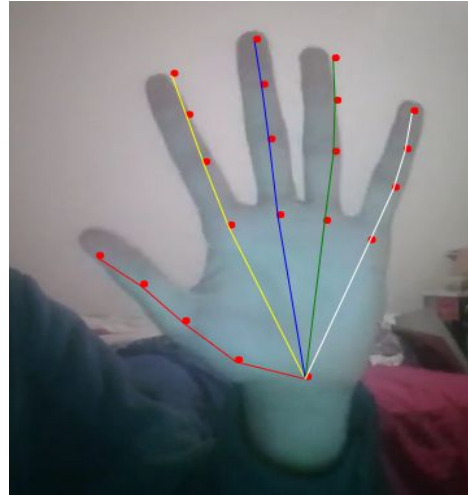
- Can easily look at fingers individually

```
▼ indexFinger: Array(4)
  ► 0: (3) [524.450620297501, 194.32627496447117, -16.261219024658203]
  ► 1: (3) [530.1228842656604, 225.04166002971678, -21.8740291595459]
  ► 2: (3) [534.5683165905356, 245.6213560813879, -24.618398666381836]
  ► 3: (3) [538.0665343191665, 262.3474677768197, -25.820755004882812]
    length: 4
    __proto__: Array(0)
▼ middleFinger: Array(4)
  ► 0: (3) [504.4905698470386, 194.578649947193, -14.893806457519531]
  ► 1: (3) [504.5600714992764, 230.2683289004499, -21.18841552734375]
  ► 2: (3) [505.63646940124465, 253.5207848003696, -22.67425537109375]
  ► 3: (3) [507.00126850425136, 271.9835807823637, -23.213748931884766]
    length: 4
    __proto__: Array(0)
▼ palmBase: Array(1)
  ► 0: (3) [507.9495722093551, 128.60076479105402, -0.00041240453720092773]
    length: 1
    __proto__: Array(0)
▼ pinky: Array(4)
  ► 0: (3) [475.0461615362033, 186.16416496483987, -9.269777297973633]
  ► 1: (3) [467.2122174779822, 209.19032666346544, -12.009689331054688]
  ► 2: (3) [463.28965099991785, 224.604732671082, -13.511730194091797]
  ► 3: (3) [460.71918659861797, 238.90564487913184, -14.570355415344238]
    length: 4
    __proto__: Array(0)
▼ ringFinger: Array(4)
  ► 0: (3) [487.9406287263003, 191.77185954684728, -12.357017517089844]
  ► 1: (3) [484.3368562572555, 223.11736709423903, -17.190589904785156]
  ► 2: (3) [483.25008055769774, 245.39809599336812, -18.510799407958984]
  ► 3: (3) [483.24238532626066, 263.47910325281043, -18.999591827392578]
    length: 4
    __proto__: Array(0)
▼ thumb: Array(4)
  ► 0: (3) [533.9069325560656, 152.78883276639914, -1.4190351963043213]
  ► 1: (3) [550.0803472419075, 177.33616611187827, -4.09717321395874]
  ► 2: (3) [560.3668451999756, 201.3174221334839, -5.182585716247559]
  ► 3: (3) [572.4152684617478, 221.90291126493398, -5.639604091644287]
    length: 4
    __proto__: Array(0)
```



Visualizing Data

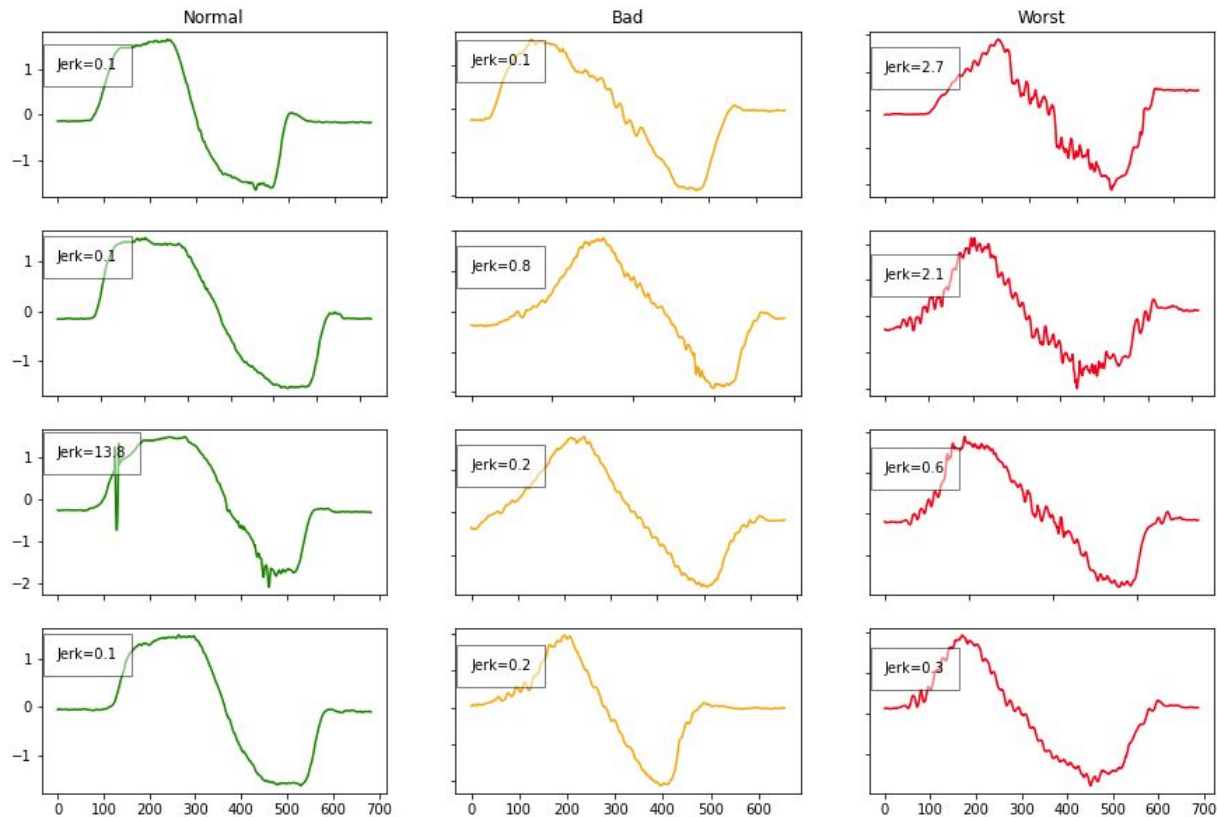
- We can draw the hand and track its movements in 3D as opposed to a video overlay



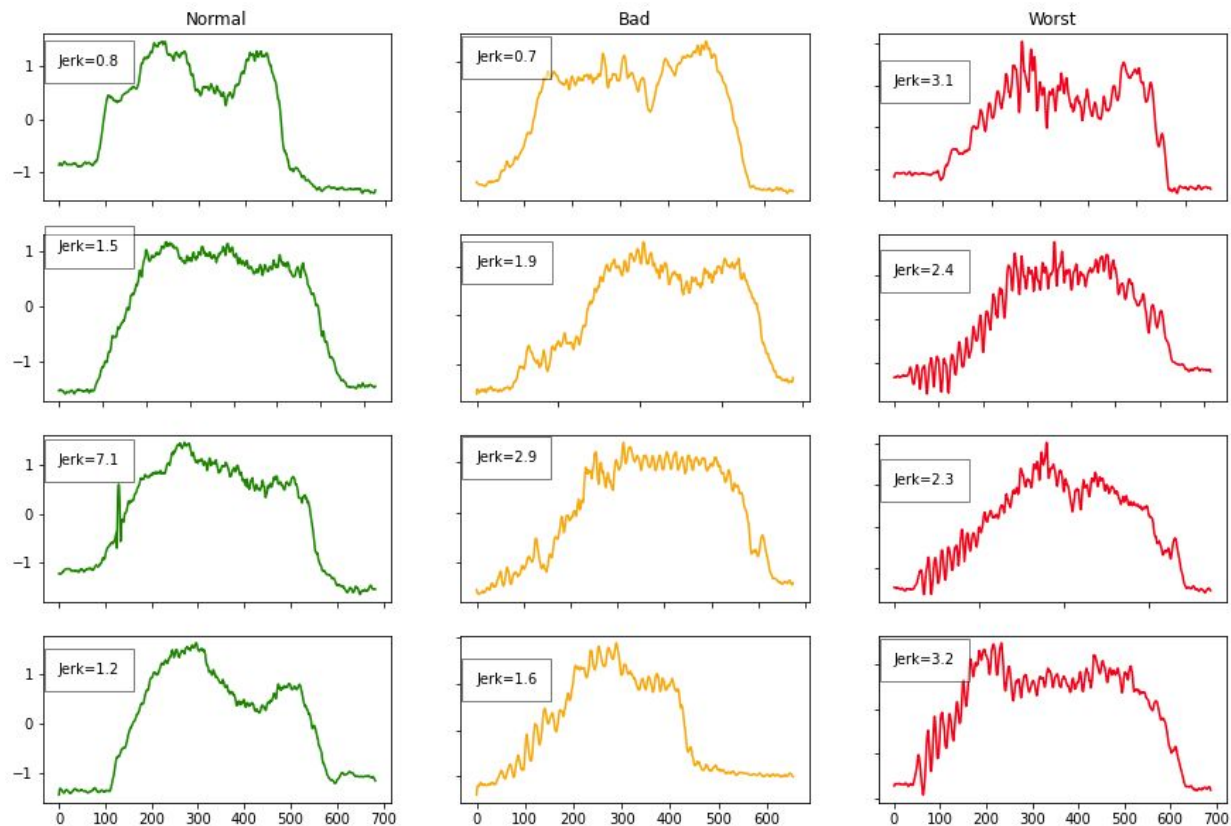
Measuring Exercise Performance

- Task: Move ball across table
- Recorded 3 sets of videos ~10 seconds long
 - Normal: smooth movement, “healthy control”
 - Bad: slight tremors
 - Worst: severe tremors
- Performed pose estimation for all videos
- Normalized Jerk Score for each keypoint
- DTW distance for trajectory comparison

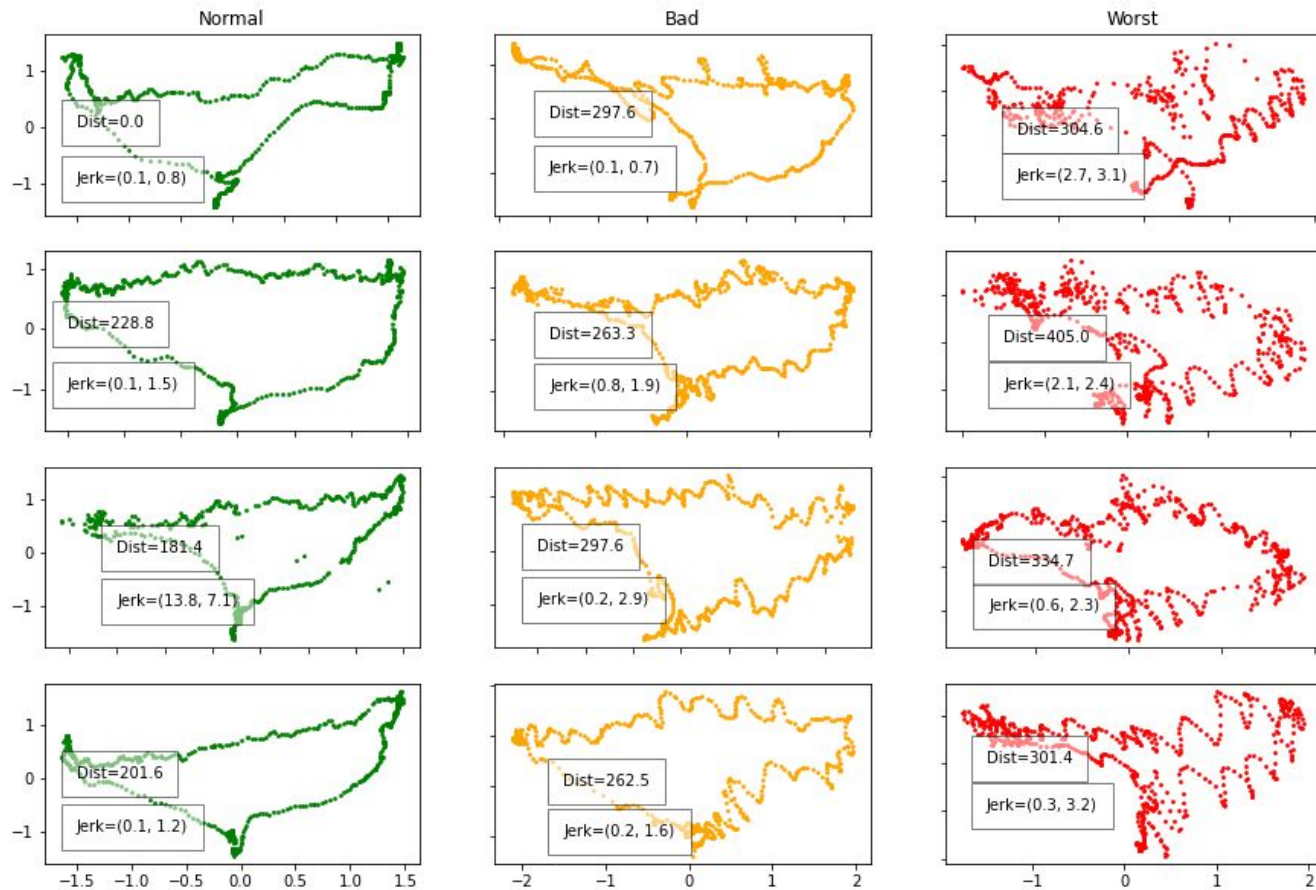
Wrist Keypoint X Values



Wrist Keypoint Y Values



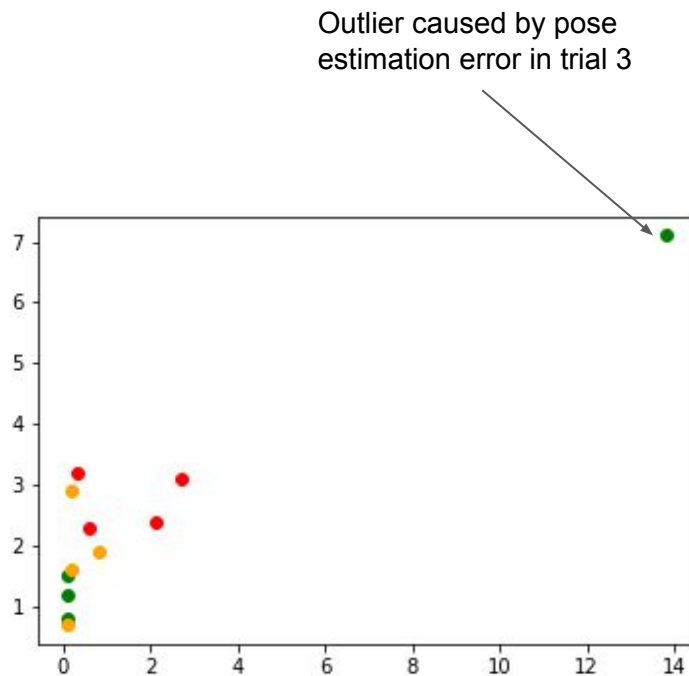
Wrist Keypoint Trajectory



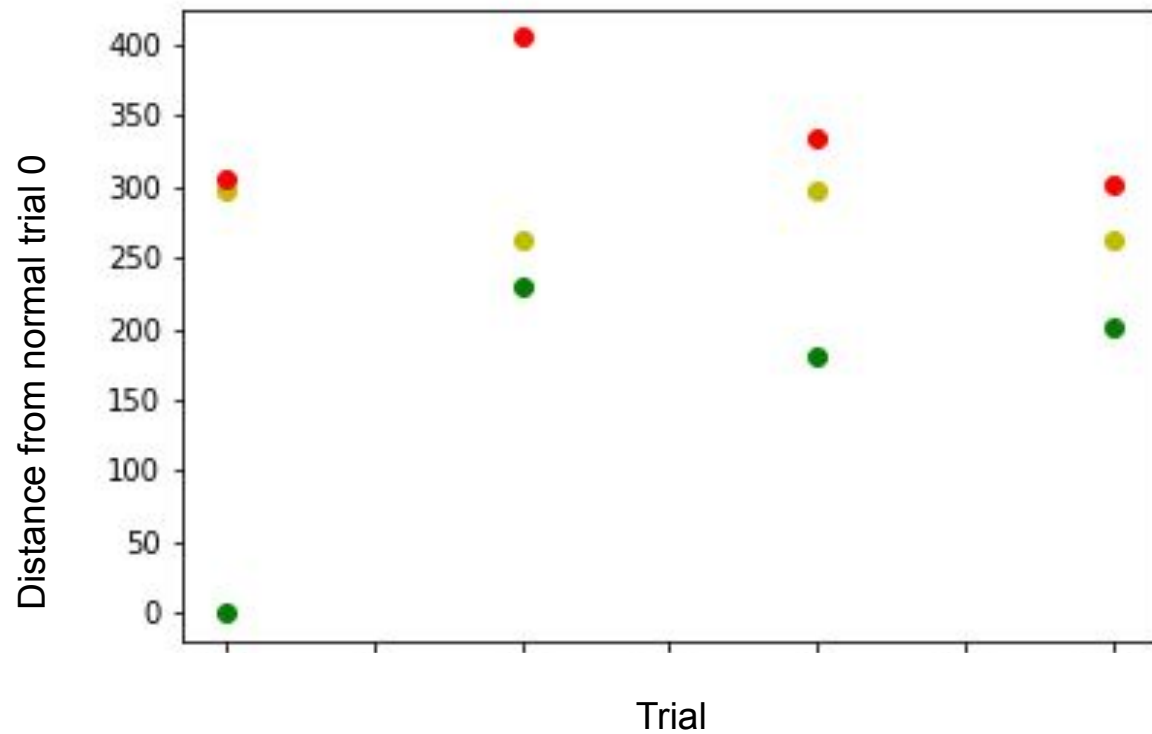
Jerk Score Data

- Jerk is computed via numerical differentiation of trajectory data
- Normalized Jerk:

$$NJ = \sqrt{\frac{1}{2} \int j^2(t) * (\text{duration}^5 / \text{length}^2) dt}$$



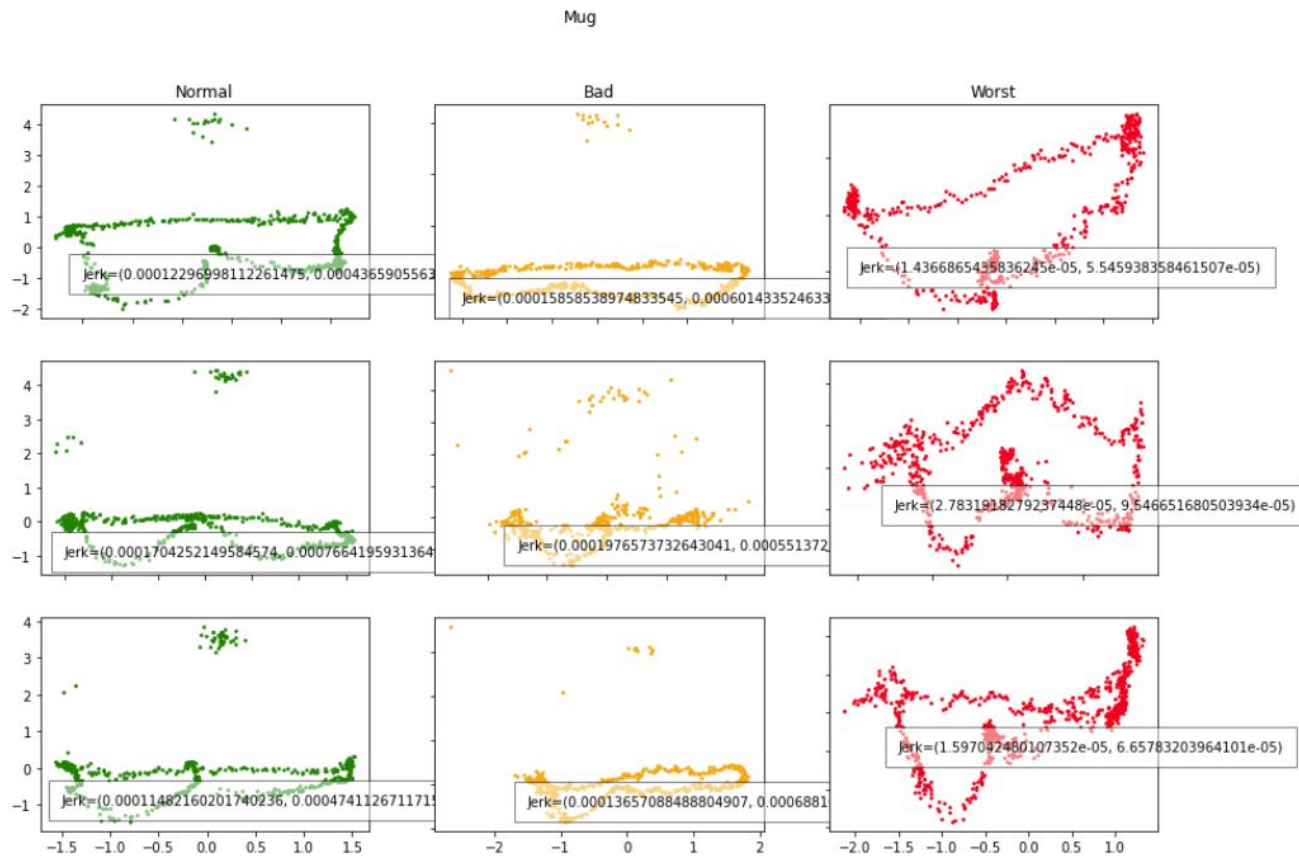
DTW Distance Data



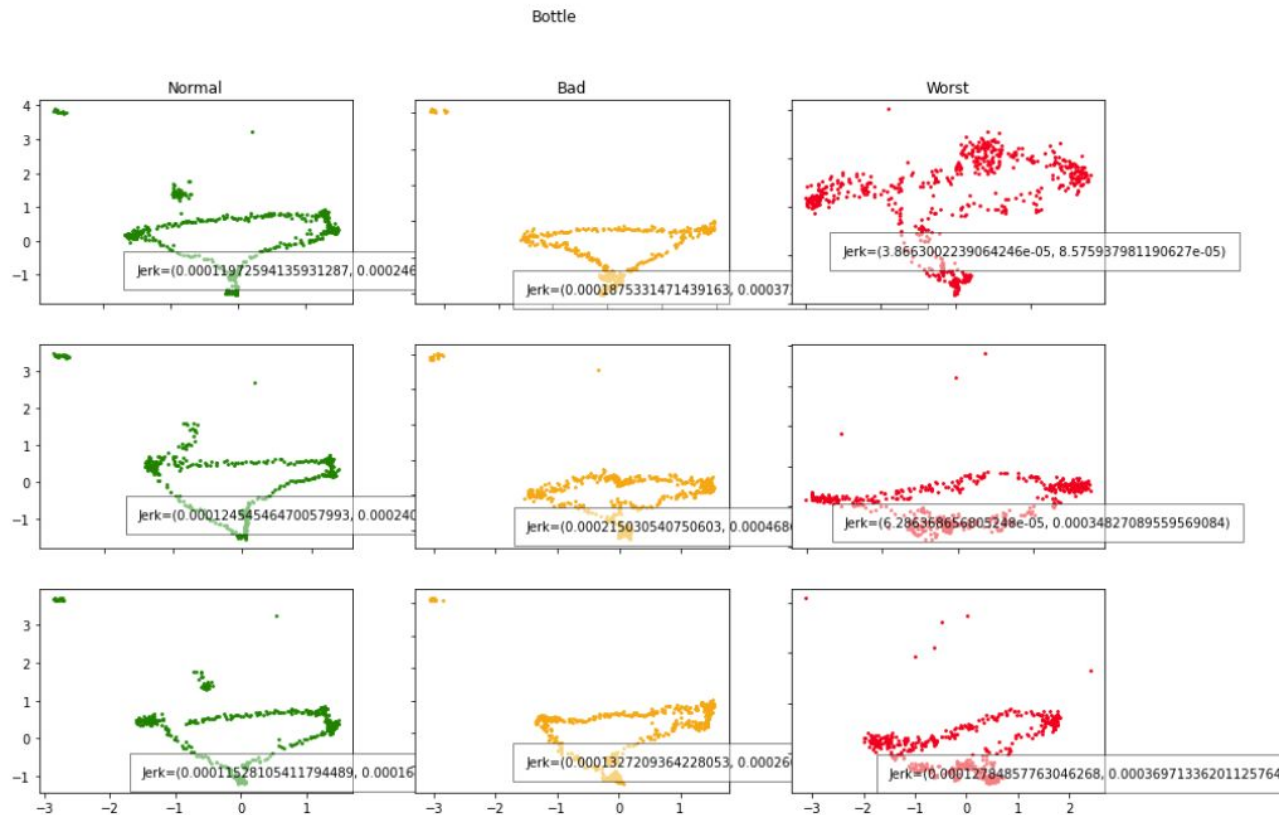
Evaluating Additional Exercises

- Tasks
 - Hook Grasp: Pick up a mug and move it across the table
 - Unscrewing: Remove the cap from a bottle and place it on the table
 - Cylindrical Grasp: Pick up a water bottle and move it across the table
- Data Collected
 - 9 videos for each exercise, separated into groups of 3 based on quality
 - pose estimation applied to video

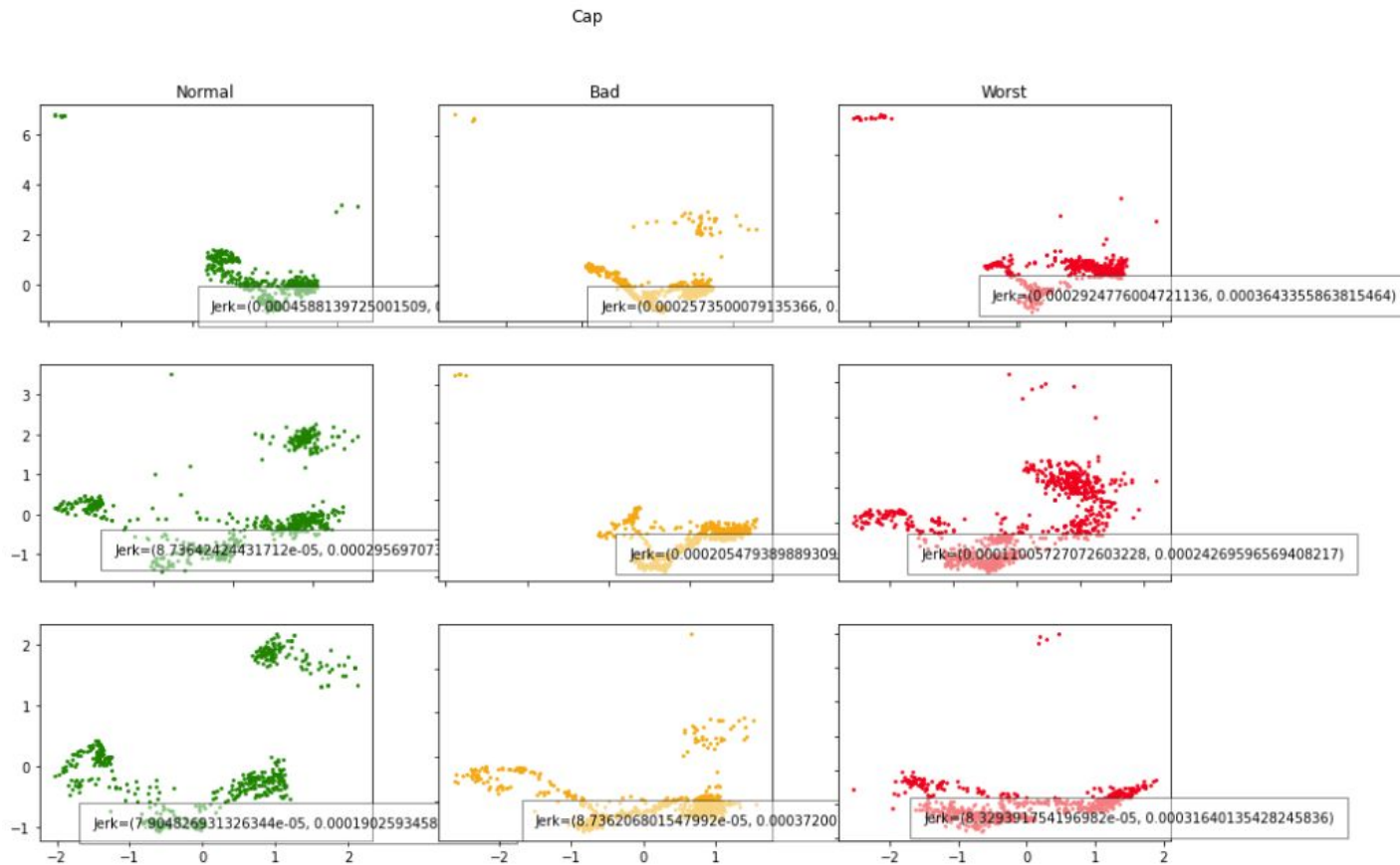
Wrist Keypoint Trajectory



Wrist Keypoint Trajectory

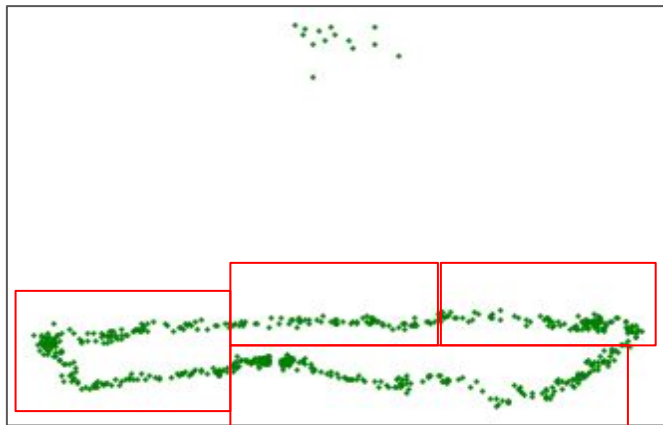


Wrist Keypoint Trajectory



Splitting Input

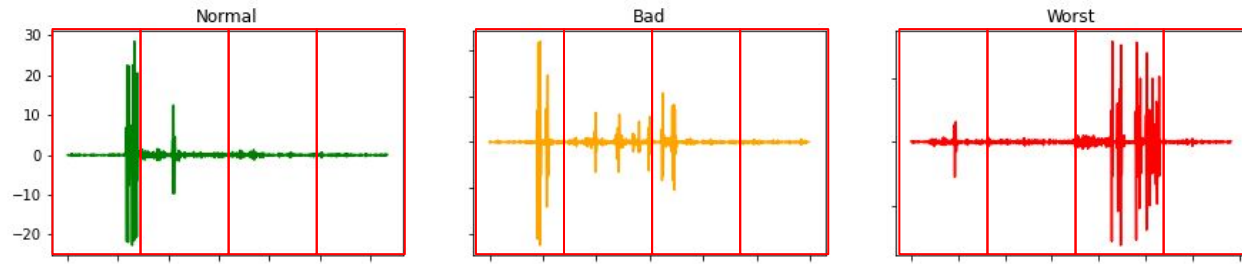
- Jerk Score is currently calculated across entire video
 - It may be useful to obtain the jerk score for certain segments, such as when a patient has just picked up the object.
- Data is temporally ordered
 - Splitting each video into K equal parts, then calculating NJS of each part individually.
 - Videos are approximately the same length



$K = 4$
Each rectangle will have its own NJS

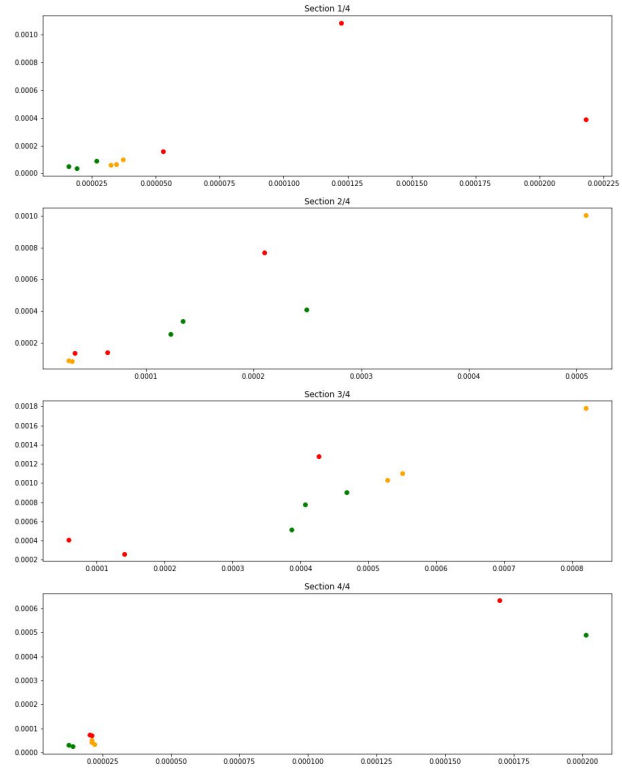
Splitting Input

- X-Coordinate Jerk Plot

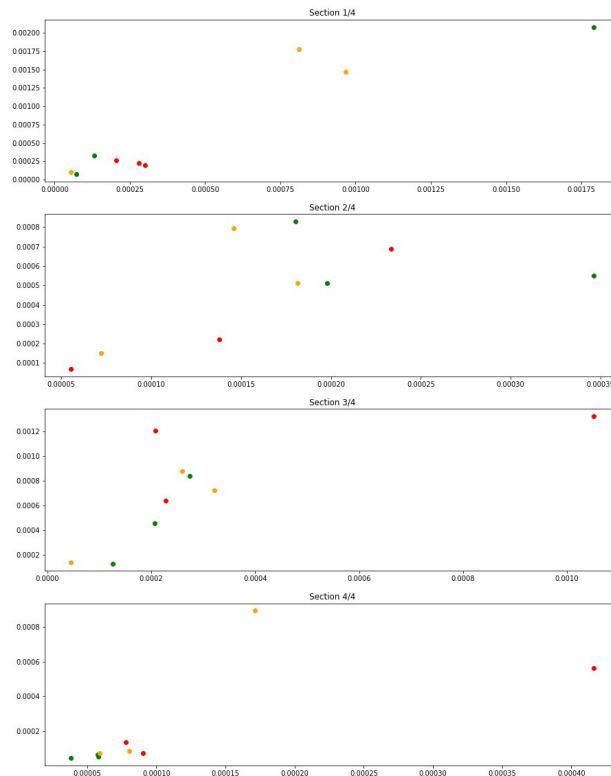


- Collected data across exercise and quality groups

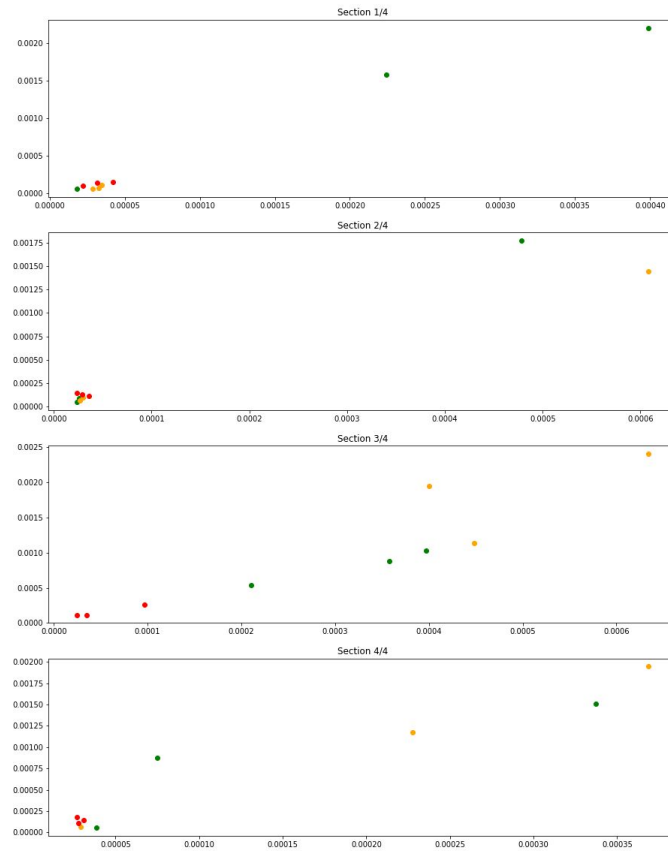
Sectional Jerk Scores - Bottle



Sectional Jerk Scores - Cap

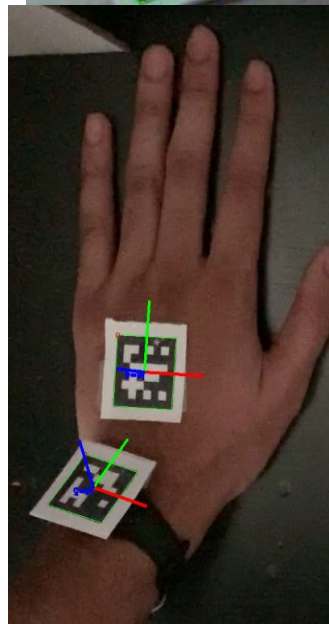
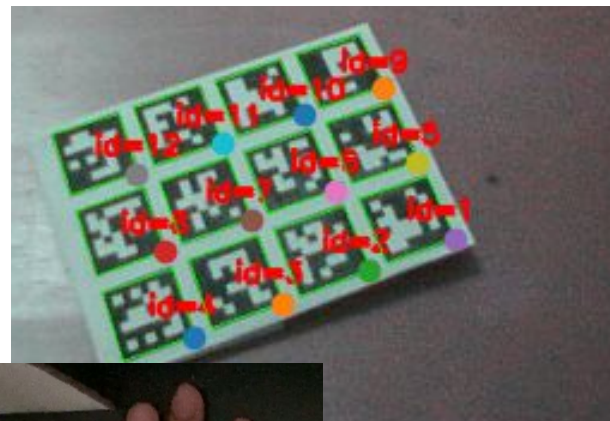


Sectional Jerk Scores - Mug



Aruco Markers

- Printable markers that allow for robust tracking
- Corners of each tag are detected
 - relative 3D pose can be approximated



Aruco Markers

- Two markers placed
- 3 cases of simple movement
 - saved (x,y) pixel locations for each marker

