Feature-Extraction-v1.2

API Contents:

Scripts:

- **KeypointExtractor.py :** It generates a **Keypoints.txt** file for each video of each exercise. Each line of the Keypoints.txt file represents a frame of the video. Each frame consists of 33 key points extracted using MediaPipe. Each keypoint has the following structure: "[ID, x, y, z, visibilityOfKeypoint]". **MediapipePose.png** shows the mapping of keypoint IDs to the actual body joint locations.
- **FeatureLoader.py :** This modifies the **EssentialFeatures.csv** and **NonEssentialFeatures.csv** files*.* It is used to add or delete or view added features for a specific exercise. Two types of features are supported, namely Essential-features and Non-Essential-features.
  - Essential features affect the frame selection process i.e. the frames for which some essential feature(s) can't be calculated (can be due to low MediaPipe visibility) will be discarded by the frame selection process implemented in the FeatureExtractor.py.
  - Non-Essential features don't affect the frame selection. In a frame, if some non-essential features can't be calculated then we use a default value (i.e. 0) for those features.
- **FeatureExtractor.py :** It creates a "***Features.txt*"** file, after performing frame selection and extracting the features that we had loaded earlier, for each video of the exercise. Each line of the Features.txt file represents a frame of the video. Each frame contains a list of features. The way a feature is represented here is as follows: [<Feature descriptor>, [<Feature values>]].
  <u>Note</u>: *The feature descriptors in the Features.txt file can be different from the feature descriptors used in the EssentialFeatures.csv and NonEssentialFeatures.csv files.*
- **datasetHandler.py :** It creates the dataset for the exercise (stored in a file named *"**master_dataset.txt**"*. Each line of the dataset represents a video and has the following syntax: "PersonID:SequenceLength:Score:SequenceOfFeatureVectors".
  <u>Note</u>: *There can be cases where all frames of a video are discarded because some essential feature(s) couldn't be calculated for any frame of the video. In such cases the SequenceOfFeatureVectors would be an empty list and the SequenceLength would be 0. Thus*

such videos are not included in the final dataset (for Feature-Extraction-v1.2 or later).

- **paddedData.py :** For a particular exercise, it generates the following:
  - tensor containing the (N,L,D) data as **padded_sequences.pt**
    - N: Number of videos of the exercise.
    - L: max(number of frames present in each video of the exercise).
    - D: Number of features (essential and non-essential features combined).
  - tensor containing the corresponding score values as **scores.pt**
  - tensor containing the corresponding sequence lengths as **sequence_lengths.pt**
  - numpy array containing the corresponding personIDs as **person_IDs.npy**

Modules:

- **FeatureTemplates.py :** Contains classes that model different types of features. Also contains classes that support basic arithmetic operations between compatible types of features.
- **PoseDetector.py :** The main purpose of this module is to extract the 33 keypoints for each frame of the input video using OpenCV and MediaPipe.

Feature Syntax and Semantics:

A feature is divided into three segments as follows:
<Dimension><Type><Parameter list>
This is the general structure of a feature descriptor.

- **Dimension:** This can be either 2 or 3. It represents the dimension of the keypoints used to calculate the feature. The feature classes that support feature calculation using 2D keypoints have been built so far.
- **Type:** Primarily 5 types of features can be calculated:
  - **D:** Distance based features
  - **K:** Raw keypoints (each keypoint is a 3-tuple (for 2D feature) containing x & y coordinates and the visibility)
  - **A:** Angle based features
  - **V:** Velocity based features
  - **OPT:** Operation between compatible types of features. This is also modelled as a feature.

Note: The “type" field of the feature descriptor is not case sensitive.

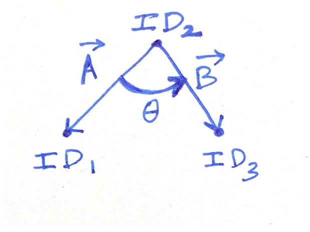
- **Parameter list:** The remaining part of the feature descriptor is the parameter list. It can contain only the keypoint IDs used to calculate the feature, or it can also include certain functions like “mid-point”, “multiplication with a scalar value” etc.
  Also, the parameter list can contain entire feature descriptors of other features. This is mainly used to represent operations or nested operations between compatible types of features.

A keypoint ID in the parameter list can be represented in either of the two ways as follows:

- ID1 : Represents the ID (can be any integer from 0 to 32) of some mediaPipe keypoint.
- m, ID1, ID2 : Represents the mid-point of two mediaPipe keypoints with identifiers ID1 & ID2 respectively.

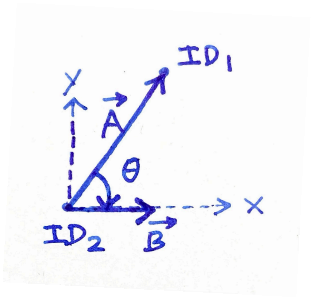
The different types of feature descriptors that the API currently supports are as follows:
(For simplicity, mid-points are not used in the general notations of the feature descriptors)

1. **Distance based:**
   - 2, D, ID1, ID2 :
     - Distance between the keypoints ID1 and ID2.
     - Return type: [distance]
   - 2, D, ID1, ID2, ID3, ID4:
     - Calculates the following ratio:
       (distance between ID1 and ID2)/(distance between ID3 and ID4)
     - Return type: [ratio]
2. **Keypoint based:**
   - 2, K, ID1 :
     - Raw keypoint with identifier ID1.
     - Return type: [x-coordinate, y-coordinate, visibility of keypoint]
3. **Angle based:**
   - 2, A, ID1, ID2, ID3, <directionFlag> :
     - Calculates the angle between vector A and vector B as shown in figure below:



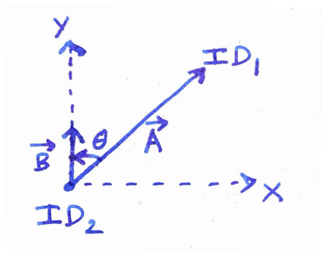
- Return type:
  - If directionFlag is “d”, then return type is: [magnitude of angle, unit vector in the direction of crossProduct(A,B)]
  - If directionFlag is “nd”, then return type is: [magnitude of angle]

- 2, A, ID1, ID2, x, <directionFlag> :
  - Calculates the angle between vector A and vector B as shown in figure below: (vector B represents positive X direction)



  - Return type:
    - If directionFlag is “d”, then return type is: [magnitude of angle, unit vector in the direction of crossProduct(A,B)]
    - If directionFlag is “nd”, then return type is: [magnitude of angle]

- 2, A, ID1, ID2, y, <directionFlag> :
  - Calculates the angle between vector A and vector B as shown in figure below: (vector B represents positive Y direction)



  - Return type:
    - If directionFlag is “d”, then return type is: [magnitude of angle, unit vector in the direction of crossProduct(A,B)]
    - If directionFlag is “nd”, then return type is: [magnitude of angle]

4. **Velocity based:**

- 2, V, ID1 :
  - The instantaneous velocity of keypoint with identifier ID1
  - Return type: [x-coordinate of velocity, y-coordinate of velocity]
- 2, V, ID1, ID2 :
  - The angular velocity of the line segment joining ID1 and ID2, with respect to the mid-point of the line segment.
  - Return type:

    - Let “mag” be the magnitude of the angular velocity
    - Let “dir” be the unit vector in the direction given by the right hand thumb rule

  The return type is: [mag, dir]
- 2, V, ID1, ID2, ID3 :
  - The rate of increase/decrease of the angle subtended at the keypoint ID2 with time.
  - Return type:
    - Let “mag” be the magnitude of the angular velocity
    - Let “dir” be such that:
      - dir = -1 if the angle subtended at ID2 increases with time.
      - dir = 1 if the angle subtended at ID2 decreases with time.

  The return type is: [mag, dir]
- 2, V, ID1, 'r', ID2, ID3 :
  - Instantaneous velocity of the keypoint with identifier ID1, scaled by the distance between ID2 and ID3.
  - Return type: [x-coordinate of scaled velocity, y-coordinate of scaled velocity]

5. **Feature operations:**
   The general structure of a feature of type “OPT” is as follows:
   2, OPT, <FeatureDescriptor_1>, <operator>, <FeatureDescriptor_2>
   <u>Note</u>:
   - <FeatureDescriptor_1> or <FeatureDescriptor_2> can also be of type “OPT”. That is, nested feature operations are allowed.
   - <operator> can take the following values:
     - add : Adds values of <FeatureDescriptor_1> and <FeatureDescriptor_2>
     - add_<num> :
       - <num> can be any integer
       - <FeatureDescriptor_1> must be same as <FeatureDescriptor_2>
       - Adds <num> to the value of <FeatureDescriptor_1>
     - sub : Subtracts the value of <FeatureDescriptor_2> from the value of <FeatureDescriptor_1>
     - sub_<num> :
       - <num> can be any integer
       - <FeatureDescriptor_1> must be same as <FeatureDescriptor_2>
       - Subtracts <num> from the value of <FeatureDescriptor_1>
     - mul : Multiplies values of <FeatureDescriptor_1> and <FeatureDescriptor_2>
     - mul_<num> :

    - <num> can be any integer
    - <FeatureDescriptor_1> must be same as <FeatureDescriptor_2>
    - Multiplies <num> to the value of <FeatureDescriptor_1>
- div : Divides the value of <FeatureDescriptor_1> by the value of <FeatureDescriptor_2>
- div_<num> :
    - <num> can be any integer
    - <FeatureDescriptor_1> must be same as <FeatureDescriptor_2>
    - Divides the value of <FeatureDescriptor_1> by <num>
- mod : Calculates the remainder after division of the value of <FeatureDescriptor_1> by the value of <FeatureDescriptor_2>
- mod_<num> :
    - <num> can be any integer
    - <FeatureDescriptor_1> must be same as <FeatureDescriptor_2>
    - Calculates the remainder after division of the value of <FeatureDescriptor_1> by <num>

Dataset Structure:

The dataset for a particular exercise is stored in a file named ***“master_dataset.txt”***. Each line of the dataset represents a video and has the following syntax: “PersonID:SequenceLength:Score:SequenceOfFeatureVectors”.

- **PersonID:** Identifier of the person performing the exercise in the particular video.
- **SequenceLength:** The number of feature vectors/frames for the particular video (after frame selection).
- **Score:** The ground truth clinical total score (cTS) value for the particular video.
- **SequenceOfFeatureVectors:** This is the sequence of feature vectors/frames for the particular video (after frame selection). For each feature vector/frame, the features are in the same order as they are present in the **EssentialFeatures.csv** followed by the **NonEssentialFeatures.csv** file. For further information about the order or the semantics/descriptors of the features present in each feature vector please refer the **Features.txt** file of any video of the particular exercise.