Biomechanical Descriptor Extract

Overview: This function uses the stride data from VAII\_kinematics\_full to get biomechanical descriptors such as Pelvic tilt variance, Joint ROM, and more. By using the function correctly, it should be easy to add new descriptors to the function outputs. This document goes over the inputs/outputs, how it works, and how to use/add to it.

Function Inputs and Outputs

This section outlines the function inputs and outputs to have an expectation of what is used and generated.

Function Inputs:

name\_cell\_l/r: List of field names of extracted data from Vicon. A field describes the type of data, such as force, joint angle, joint moment, etc.

fod\_strides\_l/r: Processed Vicon data for each field. Data is shaped F x S x n\_interp, where F is the number of fields, S is the number of strides, and n\_interp is the data interpolated to a standard size. FOD describes the method used to segment the strides (first order differential); more information in VAII\_kinematics\_full.

swingstance\_l/r: A vector containing the indices of the change between swing and stance for each stride. The vector length is S, which is the same as the number of strides in fod\_strides\_l/r. Example: to segment a stride into stance, use data from 1:ss, where ss is the index from swingstance\_l/r.

Function Outputs:

descriptors\_swing/stance: An array of descriptor values for each epoch. Array is shaped n\_d x e, where n\_d is the number of descriptors, and e is the number of epochs.

descriptors\_names\_swing/stance: A vector of each descriptors name. The descriptors correspond to each row of descriptors\_swing/stance. Swing/stance may be different sizes since not all descriptors are used in every phase of the stride, such as Knee peak early flex. Angle only pertaining to the stance phase.

How it works:

The function itself is very linear, but there are 4 different subfunctions which handle most of the work.

Here is a diagram of the program flow.

A diagram of a computer

Description automatically generated

Detailed description of each subfunction below

Subfunctions: gen\_epochs

This functions turns the stride data into sizable chunks, called epochs. Epochs are characterized by two parameters, stepsize and interval. Stepsize is the frequency at which epochs occur. Interval is the width of the epoch, or the number of strides used per epoch. Example: stepsize = 3, interval = 10 generates epochs with start/end stride indexes as [[1, 11], [13, 23], [16, 26], …]. The function should be tailored to return epochs which correspond with MV calculation.

Function Input and Outputs

Function Inputs

fod\_strides\_l/r: stride data shaped for each field.

stepsize: frequency of epochs.

interval: width of epochs. Setting interval to 0 uses all of the data from time 0 for each epoch. Example: stepsize = 3, interval = 0 generates epochs with start/end stride indexes as [[1, 3], [1, 6], [0,9]…].

Function Outputs

epochs\_l/r: array of start/end indexes for each epoch. Array is size n\_epoch x 2, where n\_epoch is number of epochs that can fit in the data given stepsize, interval, and 2 for start/end stride indexes.

How it works

In general, the start indexes are generated Matlab’s : (colon) operator. Specifically, epoch\_idx = 1:stepsize:n\_strides. This generates stepsize spaced vector up to n\_strides. Conveniently, if the next step exceeds n\_strides, then the value will not be included, making sure there is no index out of bounds error. To generate the epoch end indexes, interval is added to the epoch start index. For the special case of interval=0, 1 is used as the epoch start index for all epochs.

Subfunctions: get\_mean\_strides

This function calculates the mean stride given the specification of epochs. It utilizes Matlabs built-in mean function to calculate the mean.

Function Inputs and Outputs

Function Inputs:

strides: stride data (same as fod\_strides\_l/r)

epochs: epoch start and end indexes. The return value of gen\_epochs.

Function Outputs:

mean\_strides: mean stride for each epoch. Array is shaped n\_fields x n\_epochs x n\_interp, where n\_fields is the number of fields of data, n\_epochs is the number of epochs, and n\_interp is the length each stride (previously interpolated in VAII\_kinematics)

How it works:

For each epoch, array slice from the epoch start stride to epoch end stride, keeping all fields and all data points. Mean the sliced data across axis 2, to get the mean stride for that epoch

Subfunctions: get\_descriptor\_funcs

This function returns a cell array of descriptor function handles, used to evaluate the epoch data and get descriptor values. This function also organizes the data fields of fod\_strides\_l/r into structs, so each field can be accessed by name instead of index.

Function Inputs:

Name\_cell\_l/r: name of each field of fod\_strides\_l/r

Function Outputs:

func\_names: Array of descriptor function names as strings

descriptor\_funcs: Cell of descriptor function handles. Used so that each function can be called in a for loop instead of explicitly. Makes adding/removing functions simpler.

field\_struct: returns a struct which has the fields of fod\_strides\_l/r organized in a tree structure to be accessed by name.

How it works:

First the program runs a subfunction called build\_struct. build\_struct turns the list of field names of fod\_strides\_l/r into a struct with appropriate fields. By using string split, each field of fod\_strides\_l/r can be organized into a struct using Matlab’s default field constructor. This method prevents functions from using the incorrect field of fod\_strides if the fields were improperly organized. Example, fod\_stride\_l(2, :, :) usually refers to the JointAngle\_LHipAngles field, but if JointAngle\_LHipAngles where moved to (6, :, :), then a descriptor function directly using fod\_stride\_l(2, :, :) would be using the wrong data. Instead, using field\_struct.Left.JointAngle.HipAngles(1) always points to the correct field of fod\_strides\_l. The field struct index does not change when defining the biomechanical descriptor functions despite adding or removing fields from fod\_strides.

Using the descriptor function names in func\_list, use str2func to convert the string into a function handle. Using this method allows functions to be called from funcs using funcs{i}(params), where i is the index of the function in funcs, and params is the function signature (which should be the same for all descriptor functions).

To add more descriptor functions, add a new function definition below the Biomechanical Descriptor Functions header and add the functions name string to func\_list, along with the stride phase usage. Stride phase usage indicates which phase of the stride the descriptor is used for. For example, early peak knee flexion moment is only calculated for the stance phase, so “stance” is used as the stride phase usage tag. The “whole” tag is used if the descriptor needs the whole stride but is not calculated in swing or stance. Using the “all” tag allows the descriptor to be calculated for all stride phases.

Subfunction: extract\_descriptors/extr\_descrptr

This function uses the mean strides for each epochs and the descriptor functions to calculate descriptor values for each epoch. The function also further segments epochs into swing/stance/all phases, since descriptors in each phase of the stride are useful.

extract\_descriptors is the wrapper for extr\_descrptr, which is run for swing/stance/all.

Function Inputs:

funcs: cell array of function handles from get\_descriptor\_funcs

fields\_struct: struct of fields of fod\_strides to be accessed by name(field . notation)

strides\_l/r: mean stride data for each epoch,

swingstance\_l/r: index of swingstance change point

name: indicates which phase of stride to calculate descriptors for. Choose from swing, stance, all

Function Outputs:

descriptors: array of descriptor values. Shaped n\_d x n\_epoch, where n\_d is the number of descriptors/descriptor functions and n\_epoch is the number of epochs

name\_list: list of names of descriptor functions used for the stride phase indicated by name.

How it works:

First, the function finds which descriptor functions to use based on the phase of the stride determined by name. funcs\_use is an array of indexes for descriptor functions to use. This includes descriptors pertaining to the stride phase name, along with descriptors indicated by the “all” tag.

Next, an anonymous function is determined based on the name. The anonymous function just returns a linspace vector based on the interval.

Next, the strides\_l/r data is sliced and interpolated at the same time using slicer\_interp function. The function first array slices the data into the correct stride phase based on the phase anonymous function, then uses interp1 to interpolate the data to n\_interp points. This process is similar to normalizing strides to % stride.

Next a for loop is called on each epoch and on each descriptor function in determined in funcs\_use. Because funcs is a cell array of function handles, each descriptor function can be called using func{funcs\_use(i)}(parameters). If more functions are added to funcs, the for loop will automatically handle the new function calls in the for loop. The function then returns an array descriptors which is shaped n\_dscrptrs x n\_epochs which is each biomechanical descriptor for each epoch.