Transfer Function Documentation

for

Automated Spinal Segmentation and Remote Monitor Calibration for Surgical Assessment

Group 8: Arijit Nukala, Jonathan Young, Damiano Marsili

Mentors: Evan Haase, Antony Fuleihan

May 11, 2023

Contents

1 Transfer Function

1.1 Purpose

The transfer function utilizes patient IMU data to calculate the patient's spinal range of motion (ROM). Our project iterates upon previous attempts to calculate the spinal ROM by integrating IMU segmentation data from monocular video to introduce an optimization function to improve ROM calculation accuracy.

1.2 Project Scope

Our project can be divided into three components.

- The first component involves training an image segmentation model to accurately calculate bounding boxes for patient IMUs in monocular video. This component also utilizes the bounding boxes to estimate a spinal curvature function for input into the transfer function. This component is documented separately.
- The second component involves implementing a transfer function primarily derived from the paper ”A novel IMU-based clinical assessment protocol for Axial Spondyloarthritis: a protocol validation study.” [1] This function, implemented in Matlab, calculates spinal ROM data given IMU data as a csv input. We will be modifying the code to incorporate the IMU image segmentation based optimization.
- The third component of the project requires AWS to integrate the model, transfer function, patient access, patient data, and provider access on the cloud. This will enable patients to upload recorded IMU and monocular video data and allow clinicians to review spinal ROM data from the patient.

1.3 Design

The code used for the transfer function is pulled from Franco et al [1]. The transfer function is implemented in Matlab and takes in IMU data as a csv file. The transfer function outputs spinal ROM data as a csv file. The transfer function is modified to incorporate the IMU image segmentation based optimization. The transfer function is then integrated into AWS.

QUAL_AVA_analysis

This folder contains all files needed to run the main transfer function portion. The main file is `QUAL_AVA_analysis.m`. This file takes in the IMU data as a csv file and outputs the spinal ROM data as a csv file. The file `QUAL_AVA_analysis.m` calls the following functions:

- `Angdiff.m`: Zeros each reference frame and computes the angles between IMUs.
- `DataSegmNorm.m`: Segments angles from the motion capture system and normalizes the data.
- `EMGprocess.m`: Processes IMU EMG data to remove noise.
- `FILELIST.m`: Selects files from a specific subject and trial.
- `FINDMAX.m`: Find the maximum value of a vector corresponding to IMU data.
- `FINDMIN.m`: Find the minimum value of a vector corresponding to IMU data.
- `FindSeg.m`: Filtering data to find the start and end of a movement.
- `FindSegRelang.m`: Computing joint angles between IMUs using segmented data from movement.
- `getROM.m`: Calculates the range of motion for spinal segments.
- `getROMCleaned.m`: Analyzes ROM for mocap and IMU data and writes to ROM.mat.
- `gfit.m`: Computes goodness of fit for regression between motion capture ground truth
- `QEmatrix.m`: Visualization of the spinal ROM data.
- `QROMfunc.m`: Spinal ROM helper function.
- `QUAL_AVA_anglescomparison`: Compares the spinal ROM data from the transfer function to the ground truth data from the motion capture system.
- `quat2angle`: Converts quaternions to rotation angles.
- `quat2eul`: Converts quaternions to Euler angles.
- `quatConj.m`: Computes the conjugate of a quaternion.
- `quatinterpLoop.m`: Checks that all samples are quaternions and fixes data if not.
- `quatmultiply.m`: Multiplies two quaternions.
- `RearrDelsysCol.m`: Pulls data from the Delsys IMU csv inputs.
- `rotm2quad_mod.m`: Converts rotation matrix to quaternion.

- `RotRefSys.m`: Rotates the reference system for visualization of the IMU data.
- `STRUCTUREfilt.m`: Merges data from the Delsys IMUs and the motion capture system.
- `tilttwistang.m`: Calculates the tilt and twist angles for the IMU data.
- `XEmatrix.m`: Aligns the IMU data with the motion capture data.
- `ZeroLagButtFiltFilt.m`: Filters the IMU data with bandpass filtering for correction of data.

Method development

The code for this portion of the project was pulled from the Franco et al paper. The code was modified to incorporate the IMU image segmentation based optimization. This portion of the project was implemented in Matlab but the optimization constraints are to be defined in the future. The code was then integrated into AWS.

1.4 Optimization and Angle Calculation

1.4.1 Optimization

Flexion-extension

This constraint was calculated using the projection of the y vector projection onto the XY plane. The orientation was found with respect to the X axis using a four-quadrant inverse tangent and angles between the projection and Y axis were found by subtracting the orientation from 90 degrees.

$$\alpha_t = tan^{-1}(\frac{y_y(t)}{y_x(t)}) \tag{1.1}$$

$$\beta_t = \alpha_t - \frac{\pi}{2} \tag{1.2}$$

$$q_G^F = [cos(-\frac{\beta_t}{2}), 0, 0, sin(-\frac{\beta_t}{2})] \tag{1.3}$$

Lateral bending

This constraint was found using the angle γ between a local z axis vector and its projection onto the XY plane. A unit vector was created from the projection and rotated around the y axis by γ.

$$\gamma_t = tan^{-1}(\frac{z_z(t)}{\sqrt{z_x(t)^2 + z_y(t)^2}}) \tag{1.4}$$

$$h_{0,-\gamma} = \begin{matrix} cos(-\gamma_t) & 0 & sin(-\gamma_t) \\ 0 & 1 & 0 \\ -sin(-\gamma_t) & 0 & cos(-\gamma_t) \end{matrix} \tag{1.5}$$

A quaternion rotation representing the sensor orientation with respect to a fixed reference frame was multiplied by the rotation matrix to find the quaternion rotation of the sensor with respect to the rotated reference frame.

$$h(t) = q_F^S \otimes h_{0,-\gamma} \otimes q_F^S \tag{1.6}$$

The orientation of the projection of h_t onto the XY plane was found for all orientations.

$$\alpha_t = tan^{-1}(\frac{h_y(t)}{h_x(t)}) \tag{1.7}$$

$$\beta_t = \alpha_t - \frac{\pi}{2} \tag{1.8}$$

The quaternion rotation for which sensor orientation needs to be rotated around Z was found.

$$q_G^F = q_{BZt} = [cos(-\frac{\beta_t}{2}), 0, 0, sin(-\frac{\beta_t}{2})] \tag{1.9}$$

These constraints are adopted to other axes via:

$$h_{0,\gamma} = \begin{matrix} cos(\gamma_t) & 0 & sin(\gamma_t) \\ 0 & 1 & 0 \\ -sin(\gamma_t) & 0 & cos(\gamma_t) \end{matrix} \tag{1.10}$$

$$z_2(t) = q_F^S \otimes h_{0,\gamma} \otimes q_F^S \tag{1.11}$$

$$h_{0,0} = \begin{matrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{matrix} \cdot h_0 \tag{1.12}$$

$$z(t) = q_F^S \otimes h_{0,0} \otimes q_F^S \tag{1.13}$$

$$h_{0,-2\gamma} = \begin{matrix} cos(-2\gamma_t) & 0 & sin(-2\gamma_t) \\ 0 & 1 & 0 \\ -sin(-2\gamma_t) & 0 & cos(-2\gamma_t) \end{matrix} \quad \mathrm{cot}h_0 \tag{1.14}$$

$$z_{-1}(t) = q_F^S \otimes h_{0,-2\gamma} \otimes q_F^S \tag{1.15}$$

$$h_{0,-3\gamma} = \begin{matrix} cos(-3\gamma_t) & 0 & sin(-3\gamma_t) \\ 0 & 1 & 0 \\ -sin(-3\gamma_t) & 0 & cos(-3\gamma_t) \end{matrix} \quad \mathrm{cot}h_0 \tag{1.16}$$

$$z_{-2}(t) = q_F^S \otimes h_{0,-3\gamma} \otimes q_F^S \tag{1.17}$$

1.4.2 Angle Calculation

Joint angles were calculated using the following equations where F is flexion and extension angle, L is lateral bending angle, and T is axial rotation angle.

$$\theta = \tan^{-} 1 \frac{x_y \sin\theta + x_x \cos\theta}{x_z} \tag{1.18}$$

$$\phi = \tan^{-1} \frac{x_y \sin\theta + x_x \cos\theta}{x_z} \tag{1.19}$$

$$F = \phi \cos\theta \tag{1.20}$$

$$L = -\phi \sin\theta \tag{1.21}$$

$$T = \tan^{-1} \frac{z_z \sin\theta - y_z \cos\theta}{-z_z \cos\theta - y_z \sin\theta} \tag{1.22}$$

Bibliography

[1] L. Franco, R. Sengupta, L. Wade, and D. Cazzola, "A novel imu-based clinical assessment protocol for axial spondyloarthritis: a protocol validation study," *PeerJ*, vol. 9, p. e10623, 01 2021.