Proceedings of the 2019 IEEE/SICE
International Symposium on System Integration
Paris, France, January 14-16, 2019

# Real-time musculoskeletal visualization of muscle tension and joint reaction forces

Vincent Samy, Ko Ayusawa and Eiichi Yoshida[1]

*Abstract*— **This paper presents a novel software that visualizes the physical burden of human body during movements. Its main objective is to support factory workers by monitoring the risk of physical health problems like low back pains. To achieve the goal, the software utilizes wearable sensors like IMUs to realize the measurement at a work-site. Several physical information like joint angles, joint torques, muscle tensions, joint reaction forces can be obtained by real-time musculoskeletal computation. The musculoskeletal information can be plotted and recorded by the visualization interface which is integrated to an ergonomic assessment software DhaibaWorks.**

## I. Introduction

Several jobs in factories are still difficult to be automated and have to be performed by human workers, while the workers are facing with muscular fatigue and physical health problems in long term. Several studies have shown musculoskeletal disorders, especially low back pain, in workers [1], [2]. The low back pain is commonly caused by the load on the intervertebral discs, whose allowable limit is determined by the National Institute of Occupational Safety and Health (NIOSH) [3] and is widely used as a criteria for quantifying the risk of back pains. Since the joint reaction forces like intervertebral disc pressure are difficult to be measured directly, they are often estimated by musculoskeletal simulation [4], [5]. It is because the joint reaction forces are generated by the co-contraction of muscles rather than the external or gravitational burdens [6]. Though the off-line analysis approaches based on musculoskeletal simulation has been developed [7], the real-time estimation of joint reaction forces as well as muscle forces is therefore an important issue when monitoring the risk of physical health problems. Also intrusive methods exist [8], it is difficult to be applied in factory. The difficult part of such system is that it is impossible to directly measure muscle activities while the person is working.

This paper presents a novel software for monitoring factory workers that can be used to detect the risk of physical health problems. The developed system can provide a non-invasive way of estimating workers' muscle activities and joint reaction forces in real-time. Some recent studies report the real-time system of estimating muscle forces by a camera-based motion capture system [9]. The real-time risk estimation according to the joint reaction forces due to muscle co-contractions is further investigated in the proposed system, which is not evaluated in the above studies. Since the motion capture cameras are non-portable and unsuitable for monitoring the movements of workers, the IMU based motion capture is also integrated to the real-time musculoskeletal estimation (see Fig. 1).

## II. Software overview

The main objective of the software is to provide information about intradiscal disc pressure in a non-invasive way. Thus, the software will be able to alert about dangerous posture a worker could have while performing his tasks. Such system requires a tool-chain that starts from measurement of the human body and end at the muscle tension and the joint reaction forces. As a worker will move all around the factory, portable sensors with a wireless system seem to be the best choice as measurement tools. The system also need to visualize the status of the worker in real-time to show the current risk during the task. In this paper, the system is connected to an ergonomic assessment support software DhaibaWorks [10] for the visualization of human body. The flows of the software works as follow:

1) Receive data from sensor and extract useful information;
2) Perform an inverse kinematics (IK) of a musculoskeletal model to compute joint angles [11];
3) Compute the derivatives of the joint angles;
4) Compute the muscle tensions, joint torques and joint reaction forces through inverse dynamics computation (ID) of the musculoskeletal model [7];
5) Output the model visualization into DhaibaWorks;
6) If asked by the user, plot data in real-time;
7) If asked by the user, record the data in a csv or binary file.

In step 1), The software gets as input either motion capture markers from Cortex[1] system or XSens IMU sensors[2]. Then, step 2) computes IK from either orientations or positions, in both cases the sensors positions on the Human body is known in advance. *In fine*, step 4) solves an inverse dynamics to output wire tensions, joint torques and joint reaction forces.

In this system, the human musculoskeletal system is modeled as a wire-driven multi-body system [5]; the skeletal system is a set of grouped bones each of which is treated as a rigid body, and the musculotendon network is modeled as the wires without mass. To be able to perform IK and ID

This research was partially supported by the New Energy and Industrial Technology Development Organization (NEDO).

[1]V. Samy, K. Ayusawa, and E. Yoshida are with CNRS-AIST JRL (Joint Robotics Laboratory), UMI3218/RL, Tsukuba, Ibaraki, Japan. Corresponding Author: V. Samy `vincent.samy@aist.go.jp`

[1]Motion capture analysis software:
`https://www.motionanalysis.com/cortex/`
[2]XSens MTw Awinda software:
`https://www.xsens.com/products/mtw-awinda/`

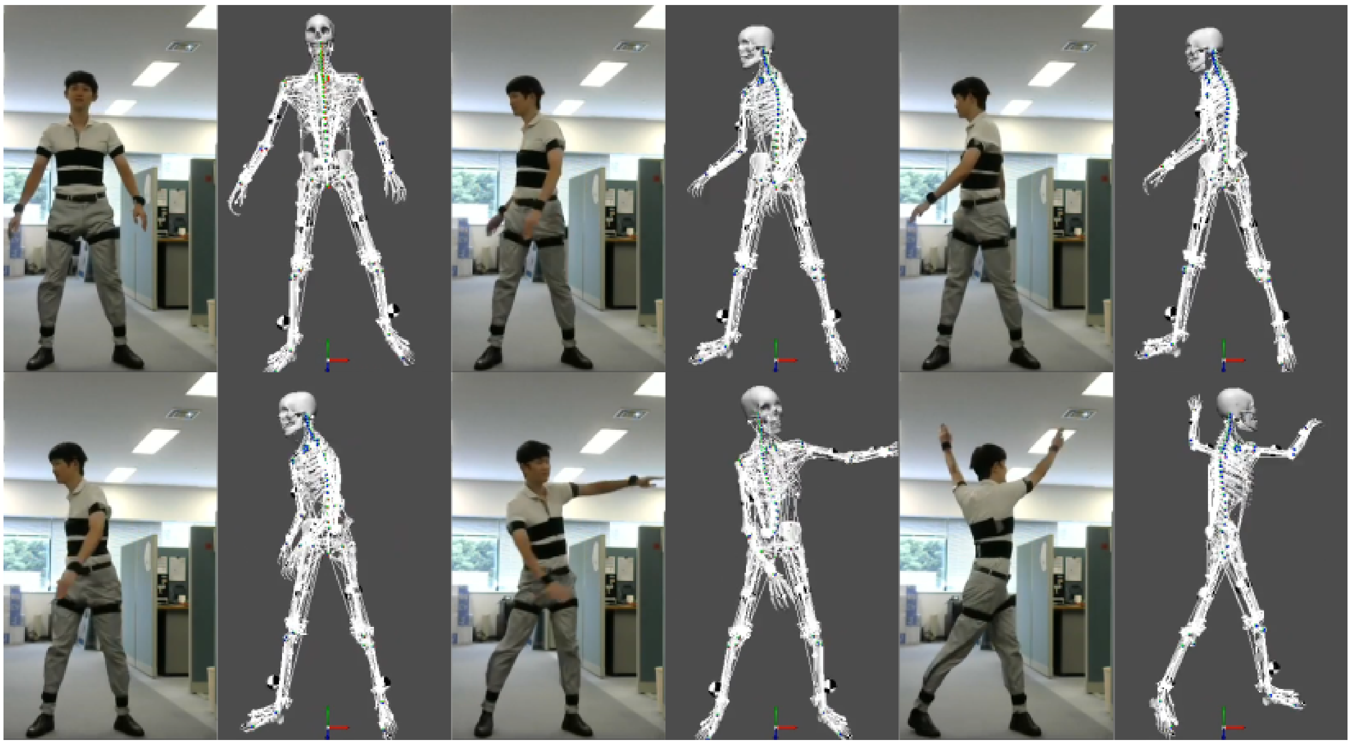


Fig. 1. Snapshots of a human wearing XSens sensor and the software model reconstruction.

computations in real-time, a simplified muscle model based is used Fig. 2.

| Nr joints | Nr muscles | Nr tendons | Nr Cartilages |
|---|---|---|---|
| $N_j = 59$ | $N_{mus} = 314$ | $N_{ten} = 6$ | $N_{car} = 34$ |

Fig. 2. Musculoskeletal model

## III. Program core

The software is trying as much as possible to parallelize the different computation processes to minimize the delay between the received data and the output and to be able to treat all data without missing one. The whole software uses notifications, condition variables and shared memory principles. In short, a thread can awake another one by sending it a notification (black arrow in Fig. 3). A thread is awaken when it receives a notification and if a condition called predicate is satisfied. Two or more threads share a memory chunk where data are temporary stored, each thread can copy data from/to this emplacement as showed in Fig. 4.

The software uses several type of threads to handle different cases. Plots and Data receiver threads have an internal timer depending on plot window framerate and sensors' frequency. In the stand-alone app, the plots run at 30fps. Data receiver threads convert sensor values into IK readable data values which are positions of the markers in the case of the motion capture system and quaternion orientation in the case of the XSens system. It also filters and performs estimation when needed. In case data from some sensors are missing (package lost due to the streaming for example), the newest data is sent.

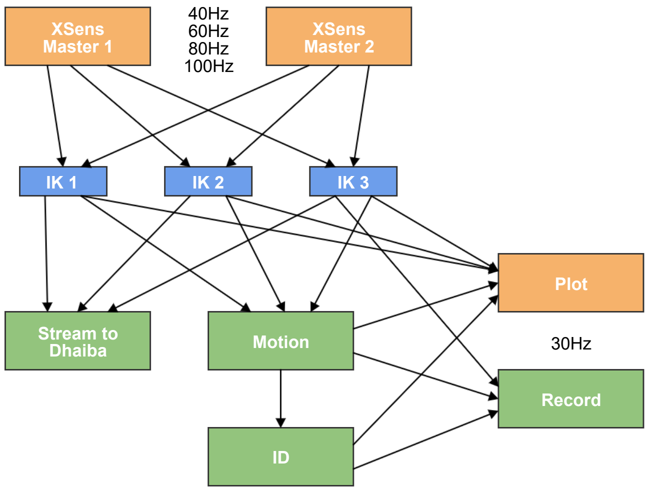


Fig. 3. Multithreading system. Rectangle are threads, arrows are between-thread communications. Orange rectangles use internal timers, blue rectangles wait to be notified but are launched one by one when all data has been synchronized. Green ones run as soon as at least one notification has been received.

To reconstruct the model 12 IMUs are needed. Unfortunately, the XSens MTw Awinda system has some constraints that needs to be handled. An XSens wireless system can only handle 6 sensors at 100Hz or 12 sensors at 60Hz which reduces the potential of the system. To overcome this problem and depending on the user's settings, the software can use several wireless master (up to 4). One data receiver thread will be spawned for each wireless master and handle

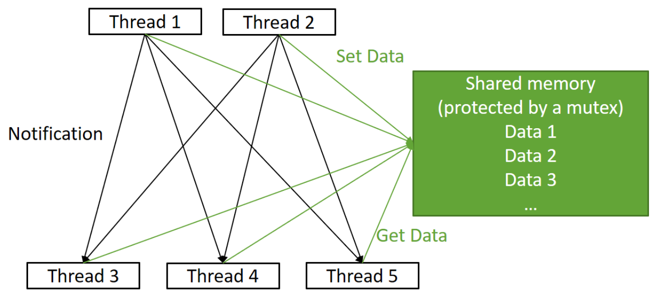


Fig. 4. Data sharing between multiple threads. A thread sends data to a shared memory chunk and then notify its child thread that data have been updated. A thread that receives a notification can either wait that all its parents have notified it or run directly.

| CPU | XSens | IK | Dhaiba | Motion | ID |
|---|---|---|---|---|---|
| Xeon 6134 | 0.06 | 9.83 | 0.66 | 0.62 | 2.04 |
| i7-7700HQ | 0.18 | 17.84 | 0.88 | 0.83 | 3.15 |

Fig. 5. Mean time of each thread over 500 samples (in ms) using XSens sensors only at 100Hz. All the data in this simulation are recorded in a csv or binary file and 6 plots for a total of 12 data are updated in real-time. Also, the joint angles are converted in DhaibaWorks' model and streamed. The measured time does not take into account the time taken by the OS to awake a thread.

6 or less sensors separately. It shares to the IK the orientation of each sensor.

IK threads can run in parallel but are launched one at a time. An IK may have a computation time long enough so that new data has already come, in that case, a second thread is launched with the new data. An IK thread starts only when all data receiver threads have sent a notification to the IK thread. Depending on the CPU, it is possible to configure the number of IK threads. Thus, for a less powerful CPU, we set the number of IK threads as equal to 4 instead of 3 for a better CPU (results shown in Fig. 5). The underlying method uses a quasi-Newton BFGS algorithm as a gradient method along a quadratic interpolation line-search algorithm.

The Dhaiba thread converts our model joint angles model $\mathrm{N}^{\mathrm{sd}}_{\mathrm{param}} = 60$ into Dhaiba joint angles model. A particularity of DhaibaWorks' model is that all joints are spherical joints with quaternions (but the free flyer which is a free joint). Thus the model has $\mathrm{N}^{\mathrm{dh}}_{\mathrm{param}} = \mathrm{N}_{\mathrm{j}} \times 4 + 3 = 239$ joint variables. The stand-alone app then streams the joint angles vector to DhaibaWorks. Note that if the msfc software has been compiled as a DhaibaWorks plugin, there is no stream and data are directly input into the DhaibaWorks model.

The motion thread computes the joint angles derivatives $\dot{\mathbf{q}}$ and $\ddot{\mathbf{q}}$. To attenuate the noise amplification, a ButterWorth filter coupled with polynomial interpolation are used to compute the derivatives. The ButterWorth filter parameters can be modified on-line through the application to test different values and search for the fittest ones. To receive data from the stand-alone app, a small plugin needs to be installed on DhaibaWorks side.

The ID thread computes the wire tension of size $\mathrm{N}_{\mathrm{f}} = \mathrm{N}_{\mathrm{mus}} + \mathrm{N}_{\mathrm{ten}} + \mathrm{N}_{\mathrm{car}} = 354$, the joint torques of size $\mathrm{N}_{\mathrm{dof}} = 47$ and the joint reaction forces of size $\mathrm{N}_{\mathrm{c}} = 6\mathrm{N}_{\mathrm{nfj}} - \mathrm{N}_{\mathrm{dof}} = 61$ where $\mathrm{N}_{\mathrm{nfj}}$ is the number of non-fixed joints of the system.

The optimization system of the inverse dynamics is [7]:

$$\begin{cases} \min\limits_{\mathbf{f},\boldsymbol{\tau}_j,\boldsymbol{\tau}_c} & Z_f(\mathbf{f}) + Z_j(\mathbf{f},\boldsymbol{\tau}_j) + Z_c(\mathbf{f},\boldsymbol{\tau}_c) \\ \text{s.t.} & \text{-}\mathbf{f}_{\max} \le \mathbf{f} \le \mathbf{0} \end{cases} \tag{1}$$

with

$$\begin{aligned} Z_f &= \mathbf{f}^{\mathsf{T}} W_f \mathbf{f} \\ Z_k &= (\boldsymbol{\tau}_k - J_k^{\mathsf{T}}\mathbf{f})^{\mathsf{T}} W_k (\boldsymbol{\tau}_k - J_k^{\mathsf{T}}\mathbf{f}), k = j, c, \end{aligned} \tag{2}$$

where $\mathbf{f} \in \mathbb{R}^{\mathrm{N_w}}$ is the wire tension, $\boldsymbol{\tau}_j \in \mathbb{R}^{\mathrm{N_{dof}}}$ is the joint torques, $\boldsymbol{\tau}_c \in \mathbb{R}^{\mathrm{N_c}}$ is the joint reaction forces, $J_j \in \mathbb{R}^{\mathrm{N_w} \times \mathrm{N_{dof}}}$ is the Jacobian matrix that maps the joint torques to the wire tension, $J_c \in \mathbb{R}^{\mathrm{N_w} \times \mathrm{N_c}}$ is the Jacobian matrix that maps the joint reaction forces to the wire tension, and $W_f$, $W_j$ and $W_c$ are weighting matrix. The algorithm itself has been proved viable in [7];

To be able to compute the above optimization in real-time the sparsity of the Jacobian matrices [12] has been exploited which even allows lower powerful CPU to be able to perform the computation in a short amount of time.

Lastly, two other threads are used for plotting data and recording in real-time. All data can be recorded in csv or binary file.

## IV. Experiments

We performed several experiments with a motion capture system and an XSens system. For the motion capture system, we use up to 34 markers for a full set of data and run the system at 200Hz. For the XSens system we use 12 sensors with one wireless master at 60Hz or 2 wireless masters at 100Hz. The following experiment uses 2 wireless masters and IMUs are disposed as such: i) Two per legs; ii) two per arms; iii) Two on the spine; iv) One on the chest; v) One the head. Back pain is determined when a joint reaction force of a vertebra crosses the threshold of 3400N [3].

### A. Methodology

The setup for the experiment is quite simple. Each XSens sensor has an 8-digits unique identifier to be able to be recognized it. This identifier is carry out through the software to know which sensor will be stick on which part of the body, thus it allows the software to know where each sensor is on the body. We also stick on each sensor the name (*e.g.* Left thigh, Right wrist, etc) of the body part so it is human readable and anyone can install it. First, the software needs to be started to look for all 12 sensors and to associate each sensor to a wireless master and thus to a data receiver thread. Then, the sensors are attached to the wrists, to the shoulders, to the ankles, to the thighs, to the T7 and L1 vertebra, to the chest and one to the head (Fig. 6). Once the IMUs are placed on the body, the subject needs to stand straight, as in Fig. 6, to perform an initialization procedure. This procedure will synchronize the orientation between the sensors and the model and it takes around 1s.

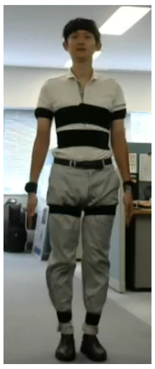


Fig. 6. Initial pose. This step is needed to synchronize the sensors to the software. The position of the IMUs and initial orientation are predetermined.

| CPU | Model update delay | Data plots delay |
|---|---|---|
| Xeon 6134 | 10.55 | 12.55 |
| i7-7700HQ | 18.9 | 22 |

Fig. 7. Delay time (in ms) between the motion and the software outputs. The model update delay is the time it takes to compute the IK and send the results visualizer. The data plots delay is the time to compute the ID and send the data to the real-time plots.

### *B. Data analysis*

We have tested the software with two different pc: a desktop with a Intel(R) Xeon(R) Gold 6134 CPU at 3.20GHz and a notebook with a Intel(R) Core(TM) i7-7700HQ at 2.80Ghz. The test is done with heavy constraints *i.e.* all data are recorded in a csv file and 6 plots are updated with 2 curves on each in real-time and the model is visualized on the same pc under DhaibaWorks software. The results have been written in Fig. 5 and as expected the Xeon cpu performs better. The table also shows that without the multithreading technics, real-time performances would not be possible. Indeed, it takes 12.55ms for the Xeon cpu to process the data, whereas the sensor frequency is 10ms (100Hz), or 5ms (200Hz) for the motion capture system. For the Xeon CPU, 3 IK threads can run in parallel while for the i7, 4 IK threads are used. Overall, the i7 takes twice the times of the Xeon. Note that even if the computation is done in real-time there is still a delay between the input and the output of the software Fig. 7. So, the online visualization of the model has delay of 10.55ms to 18.9ms and the online plots have a delay of 12.55ms to 22ms depending on the platform used.

All the data can be displayed via DhaibaWorks software Fig. 8.

## V. Conclusion

This paper presents a new software of visualizing human musculoskeletal information in real-time. The software can reconstruct human joint movements from IMU sensors or a motion capture system by inverse kinematics computation. IMU sensors are running at 100Hz while the motion capture system runs at 200Hz, without loss of data. The inverse dynamics computation is then performed to estimate muscle tensions, joint torques and joint reaction forces. The computation exploits the sparsity of the Jacobian matrices to lower the computation time below 5ms even for less powerful CPU. Thanks to the multithreading technology and depending on the CPU performances, the delay time from the human motion to inverse dynamics output varies from 10ms to 25ms.

Several experiments have been conducted to test the software under different platforms and with different sensors at different rate. During the experiments back muscle tensions and vertebra joint reaction forces have been monitored. The simulated model is visible through the DhaibaWorks software along with plots that can show variables of the system.

## VI. Future work

This paper shows that real-time monitoring of workers' labor is possible. One of the main constraint of this system is the number of the IMUs and the drift that occurs with time. As the workers would need to wear the IMUs all day long, the number of IMUs becomes cumbersome. The drift in the IMU is another problem which requires the bearer to reset the axis alignment several time a day to cancel it. The next step for this research will be to reduce the number of IMUs and find a way to correct the drift.

## References

[1] B. Duthey, "Background paper 6.24 low back pain," *Priority medicines for Europe and the world. Global Burden of Disease (2010),(March)*, pp. 1–29, 2013.

[2] L. Punnett, A. Prüss-Ütün, D. I. Nelson, M. A. Fingerhut, J. Leigh, S. Tak, and S. Phillips, "Estimating the global burden of low back pain attributable to combined occupational exposures," *American journal of industrial medicine*, vol. 48, no. 6, pp. 459–469, 2005.

[3] T. R. Waters, V. Putz-Anderson, A. Garg, and L. J. Fine, "Revised niosh equation for the design and evaluation of manual lifting tasks," *Ergonomics*, vol. 36, no. 7, pp. 749–776, 1993.

[4] S. L. Delp, F. C. Anderson, A. S. Arnold, P. Loan, A. Habib, C. T. John, E. Guendelman, and D. G. Thelen, "Opensim: Open-source software to create and analyze dynamic simulations of movement," *IEEE Trans. on Biomedical Engineering*, vol. 54, no. 11, pp. 1940–1950, 2007.

[5] Y. Nakamura, K. Yamane, Y. Fujita, and I. Suzuki, "Somatosensory computation for man-machine interface from motion-capture data and musculoskeletal human model," *IEEE Transactions on Robotics*, vol. 21, no. 1, pp. 58–66, 2005.

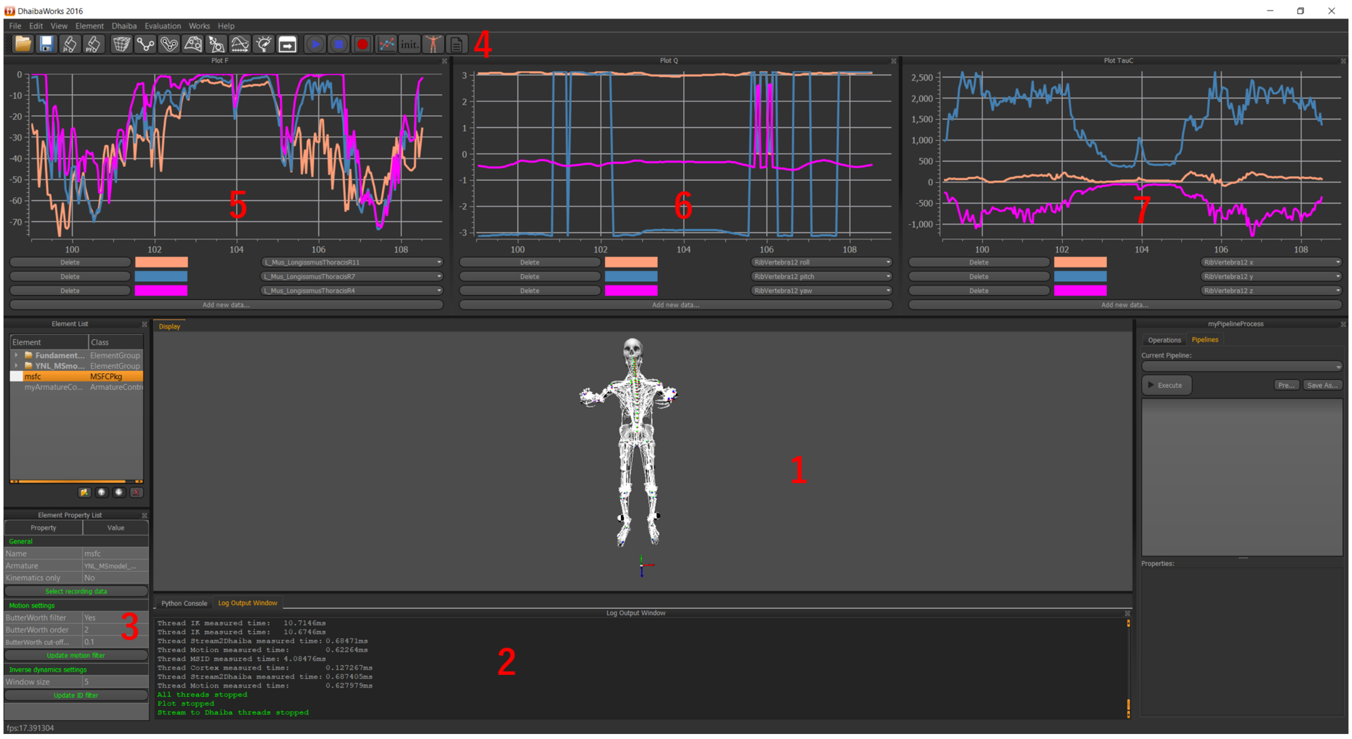
[6] I. Takahashi, S.-i. Kikuchi, K. Sato, and N. Sato, "Mechanical load of the lumbar spine during forward bending motion of the trunk - a biomechanical study," *Spine*, vol. 31, pp. 18–23, 02 2006.

[7] Y. Imamura, K. Ayusawa, and E. Yoshida, "Risk estimation for intervertebral disc pressure through musculoskeletal joint reaction force simulation," in *Engineering in Medicine and Biology Society (EMBC), 2017 39th Annual International Conference of the IEEE*. IEEE, 2017, pp. 1636–1639.

[8] H.-J. Wilke, P. Neef, B. Hinz, H. Seidel, and L. Claes, "Intradiscal pressure together with anthropometric data–a data set for the validation of models," *Clinical Biomechanics*, vol. 16, pp. S111–S126, 2001.

[9] A. Murai, K. Kurosaki, K. Yamane, and Y. Nakamura, "Musculoskeletal-see-through mirror: Computational modeling and algorithm for whole-body muscle activity visualization in real time." *Progress in Biophysics and Molecular Biology*, vol. 103, no. 2-3, pp. 310–317, 2010.

[10] Y. Endo, M. Tada, and M. Mochimaru, "Dhaiba: development of virtual ergonomic assessment system with human models," in *Proceedings of The 3rd International Digital Human Symposium*, 2014.



| Property | Value |
|---|---|
| General | |
| Name | msfc |
| Armature | YNL_MSmodel_... |
| Kinematics only | No |
| Select recording data | |
| Motion settings | |
| ButterWorth filter | Yes |
| ButterWorth order | 2 |
| ButterWorth cut-off... | 0.1 |
| Update motion filter | |
| Inverse dynamics settings | |
| Window size | 5 |
| Update ID filter | |

```
Thread IK measured time:   10.7146ms
Thread IK measured time:   10.6746ms
Thread Stream2Dhaiba measured time: 0.68471ms
Thread Motion measured time:        0.62264ms
Thread MSID measured time: 4.08476ms
Thread Cortex measured time:        0.127267ms
Thread Stream2Dhaiba measured time: 0.687405ms
Thread Motion measured time:        0.627979ms
All threads stopped
Plot stopped
Stream to Dhaiba threads stopped
```

fps:17.391304

Fig. 8. DhaibaWorks with the integrated plugin. The plugin adds several tools to the main window. It adds 1) visualization of the musculoskeletal model, 2) information into the console on the state of the software, 3) tunable parameters such as filters, 4) toolbar buttons to easily play, record, initialize the software 5-7) plots to visualize on-line state. 5) show 3 back muscles (Longissmus Thoracis 11, 7, 4), 6) is the rib vertebra 12 orientation (roll, pitch, yaw) and 7) show joint reaction forces of the rib vertebra 12 (x, y, z).

[11] K. Ayusawa and Y. Nakamura, "Fast inverse kinematics algorithm for large dof system with decomposed computation of gradient and its application to musculoskeletal model," in *Proc. of the 17th Robotics Symposia*, 2012, pp. 148–155.

[12] ——, "Fast inverse dynamics algorithm with decomposed computation of gradient for wire-driven multi-body systems and its application to estimation of human muscle tensions," in *2nd IFToMM International Symposium on Robotics and Mechatronics (11)*, 2011.