Implementation of Quaternion Based Lifting Scheme for Motion Data Editor Software

Mateusz Janiak[1], Agnieszka Szczęsna[2], and Janusz Słupik[2]

[1] Polish-Japanese Institute of Information Technology, Bytom, Poland
Mateusz.Janiak@pjwstk.edu.pl
[2] The Silesian University of Technology, Institute of Informatics, Gliwice, Poland
{Agnieszka.Szczesna,Janusz.Slupik}@polsl.pl

Abstract. Motion analysis is rapidly developing area of research. Due to availability of cheaper hardware with reasonable accuracy for motion acquisition in form of various motion controllers, dedicated mainly for gaming, there are rising new research groups interested in this topic. Proposed solutions for motion analysis have a wide range of application in medicine, sport, entertainment and security. Although motion analysis is one of the most important domains of our everyday life, there are still no good tools supporting knowledge exchange and experiments in this field. In this paper we want to introduce a possibility of implementing specialised wavelet analysis in form of the lifting scheme for motion data in quaternion representation in a data flow processing framework available in Motion Data Editor (MDE) software developed at Polish-Japanese Institute of Information Technology (PJWSTK) in Bytom (Poland). We want to show how easily custom solutions can be introduced to this general purpose data processing software. Usage of this software saves time by concentrating on rapid prototyping of new algorithms and performing experiments, skipping creation of similar solutions for various data types and algorithms.

Keywords: motion analysis, software architecture, data flow, data processing, lifting scheme, quaternions, wavelets.

1 Introduction

Nowadays new techniques are being developed and improved for motion analysis. Among them we can find tools dedicated to motion data comparison and compression. Additionally, many algorithms for motion segmentation, recognition and classification are proposed. Unfortunately, there are no tools that support motion analysis and processing. Two available products: commercial Vicon Polygon [16] and open-source Mokka [1], provide only motion data browsing and visualisation, with no data processing features. Moreover, users can not extend any of those applications to fit their particular needs, in example introducing custom data types and algorithms. To address this problem a new software called MDE was developed at PJWSTK in Bytom (Poland). This is a general purpose data processing tool, with dedicated extensions for motion analysis and medical

N.T. Nguyen et al. (Eds.): ACIIDS 2014, Part II, LNAI 8398, pp. 515–524, 2014.

applications. In this paper we want to present briefly the main features of MDE and show how they affect research work in the domain of motion analysis. As an example, the lifting scheme for multiresolution data analysis has been implemented in the MDE software. We present how particular tests can be performed and how to collect their results for noise reduction and data compression of motion data.

1.1 Motion Data Editor

The MDE software is developed at PJWSTK in Bytom (Poland). Initially it was designed to support medics in diagnosis of various motion dysfunctions. Since then it has been reorganised and refactored to become a mature tool for a general purpose data processing and analysis. The main power of the MDE software is its architecture dedicated to a well defined data processing pipeline.

MDE architecture supports flexible application extension with custom data types, their dedicated operations and algorithms. This is achieved with a specialised plug-in system allowing to fit MDE to particular users' needs, standardized their solutions around one tool and save time on implementation of similar functionality operating on different data types.

Architecture. In the MDE architecture are objects responsible for particular stages of data processing pipeline. In the first step user browse for data that to process. This is realised by *Source* objects. They are responsible for delivering data in containers (usually various file formats) to local hard disk, if required. Later, containers are unpacked with help of *Parser* objects. Loaded data are normalised and wrapped with help of *ObjectWrappers*. This is a completely new approach to uniform data handling in statically and strongly typed C++ programming language with just a small memory and performance overhead. This technique combines generic programming and run-time type information (RTTI). It outperforms functionality of different *variant* types. It allows simple, yet efficient data exchange between all application modules. After loading the data to application user might view the data with help of *Visualiser* objects, presenting data graphically at various perspectives. *Visualisers* allow to present and compare more data on a single scene. *Services* allow users to introduce new functionality to application. This allows users to introduce new algorithms to MDE or functionality completely unconnected with data analysis and processing. All those objects can be introduced to MDE through a dedicated plug-in system.

Features. As a cross-platform software, MDE provides an abstraction layer for most system specific functionality like file system management and threading. Additionally, a dedicated hierarchical *Log* system was designed to simplify notifying about application status. Dedicated managers with transactional mechanisms offer thread-safe memory operations. To control and monitor amount of threads used in application a concept of *ThreadPool* has been introduced. *Threads* are expected to perform simple operations, mainly waiting for some

events and actions to schedule their processing. For heavy computations an idea of *Jobs* and *JobManager* is proposed. *Job* represents particular computations that are scheduled to *JobManager*. *JobManager* is responsible to run *Jobs* utilizing optimally available computational resources (specified number of worker threads depending on available central processing unit (CPU)). Analysed data is very often indexed with time - this is especially the case for various physical measurements. To address problem of uniform handling data indexed with time a dedicated and generic *DataChannel* type was introduced. It allows to create illusion of data continuity in the time domain with help of various inter- and extrapolation methods. To handle efficiently memory used by the loaded data a lazy initialization mechanism is proposed.

1.2 Data Flow

The concept of processing data in form of a well defined pipeline is widely applied in CPU architecture [12,11,9,3,4]. Each instruction execution is divided into separate stages realised in dedicated modules. This allows to execute several instructions in parallel on a single CPU. To allow users simple composition of complex data processing algorithms, the concept of data flow processing framework was introduced to MDE in form of a dedicated service. It is based on a graph structure extended with elements called *Pins*. *Pins* are attached to nodes, providing nodes input and output for data. Nodes can be connected together only through compatible *Pins*. Created processing model can be saved and restored for later usage. Among nodes three groups can be defined: sources, processors and sinks (Figure 1).

Source nodes deliver new data for processing in data flow. Processing nodes consume data from attached input pins and provides new data through output pins. Sinks are used to save results of processing for further analysis. Data are

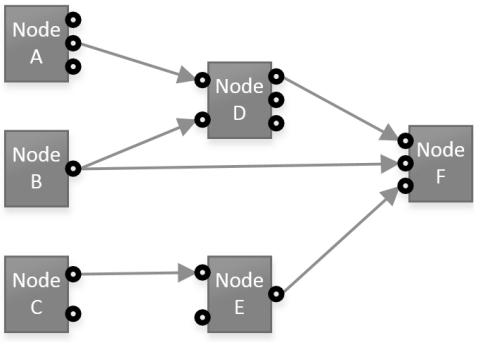


Fig. 1. Example of data flow model

considered to leave the model either when they are processed by sink nodes or when corresponding output pins are not connected. Such approach allows simple utilisation of available computational resources by handling processing logic of each node by a separate thread from *ThreadPool* and scheduling node‘s processing in form of a *Job* to *JobManager*. With such approach users have to implement processing functions and encapsulate their signatures within proper nodes, where functions‘ inputs and outputs are represented by particular *Pins* with the given data types.

1.3 Visual Data Flow

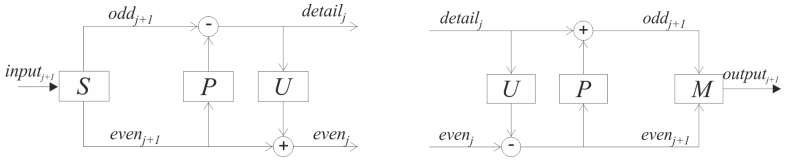
Based on data flow processing model a graphical programming environment was created to simplify creation of processing pipelines. Using drag‘n‘drop mechanism users can easily choose interesting functions and compose them together creating more specific algorithms. Those algorithms can be saved and restored for further usage. It is also possible to group several basic nodes connected together and replace them with a single virtual node. This helps to keep scene clear and maintain good overview of composed algorithm. While creating processing pipeline user is guided visually which pins can be connected together by verifying their data types compatibility.

2 Lifting Scheme and Experiments in MDE

Multiresolution tools are widely used for motion data analysis. They usually operate on Euler angles representation for rotations (orientations) [7,8,5,6,2]. In our work we are developing new algorithms based on the second generation wavelets constructed by the lifting scheme, that work on quaternion representation for rotations. Based on our previous work, where we have proposed a lifting scheme for quaternions in tangent space (computed by logarithm) using classical Bezier interpolation in $\mathbb{R}^3$ space we want to present a new approach using $SQUAD$ quaternion interpolation [15,14,13]. Using quaternion lifting scheme based on quaternion algebra we can work directly on correlated motion data. We decided to perform particular test of its application for data compression and noise reduction within MDE.

2.1 *SQUAD* Lifting Scheme

To construct proposed lifting scheme we use classical split block, where samples are divided on odd o_i^j and even e_i^j indexed samples, where j is the resolution level and i is the index of sample (Figure 2). The odd indexed input value after one step of lifting scheme is replaced by the difference (detail value) between the odd value and its prediction. The even indexed samples are updated, so that coarse-scale samples have the same average value as the input samples. Based on even indexed samples we propose prediction and update blocks for forward transform as follows:



(a) forward lifting scheme **(b)** inverse lifting scheme

Fig. 2. The lifting scheme

– predict
SQUAD interpolation uses four nearby even indexed samples to predict odd sample according to the equation 5.

$$o_i^j = o_i^{j+1} * squad\left(e_i^{j+1}, e_{i+1}^{j+1}, e_{i+2}^{j+1}, e_{i+3}^{j+1}, 0.5\right)^{-1} \tag{1}$$

– update
The update step is obtained in order to preserve the equality (average of the signal) as

$$e_{i+1}^j = \left(o_{i+1}^j\right)^{0.5} * e_{i+1}^{j+1} \tag{2}$$

For the inverse transform the following equations are given:

– undo-predict

$$o_i^{j+1} = o_i^j * squad\left(e_i^{j+1}, e_{i+1}^{j+1}, e_{i+2}^{j+1}, e_{i+3}^{j+1}, 0.5\right) \tag{3}$$

– undo-update

$$e_i^{j+1} = \left(o_i^j\right)^{-0.5} * e_i^j \tag{4}$$

Applying Bezier curves interpolation idea to quaternions we obtain SQUAD interpolation (equation 5) [10]. It requires four quaternions for interpolation, two of them are used as a interpolation range, with other two used to generate control points ensuring smooth and differentiable interpolation curve. Quaternions q_{i-1} and q_i are used as key frames and quaternions q_{i+1} and q_{i+2} are control points.

$$squad\left(q_{i-1}, q_i, q_{i+1}, q_{i+2}, t\right) = slerp\left(slerp\left(q_{i-1}, q_i, t\right), slerp\left(q_{i+1}, q_{i+2}, t\right), 2t\left(1-t\right)\right) \tag{5}$$

and

$$slerp\left(q_a, q_b, t\right) = q_a\left(q_a^* q_b\right)^t \tag{6}$$

Equation 7 describes how control points are generated.

$$s_i = q_i exp\left(-\frac{log\left(q_i^{-1}q_{i+1}\right) + log\left(q_i^{-1}q_{i-1}\right)}{4}\right) \tag{7}$$

2.2 Applications

We want to apply presented lifting scheme for data compression and noise reduction. In case of data compression we propose lossy algorithm. It is based on removing of wavelet quaternion detail coefficients of the specific (see 2.5) number of the highest resolutions. The detail coefficient is the difference between the odd indexed sample and its prediction. The reconstruction introduces differences to original signal because part of the data was removed. For noise reduction we are using selective soft threshold α according to the following rule:

$$q_d = \begin{cases} q_d & \text{if } \alpha < v < 1 - \alpha \\ [1, (0, 0, 0)] & \text{otherwise} \end{cases} \tag{8}$$

where $v = \frac{arccos(a)}{\pi}$ and detail is a quaternion $q_d = a + bi + cj + dk$. Such approach eliminates meaningless rotations (close to 0°, changing unnoticeable body orientation) and significant rotations (close to 360°, moving body close to its initial orientation). As rotation quaternions are always represented as a cosine of half desired rotation angle, therefore α values were limited to the given range $(0, 0.5)$.

As a measure of quality for the presented applications we introduce a distance measure for quaternion signals. For two normalised quaternions q_a and q_b we can define the distance as:

$$d_q\left(q_a, q_b\right) = \left|arccos\left(Re\left(q_a q_b^{-1}\right)\right)\right| \tag{9}$$

For two quaternion signals Q_A and Q_B with N quaternion samples, we can define the distance between them as:

$$D(Q_A, Q_B) = \frac{1}{N}\sum_{i=1}^{N}\left(d_q\left(q_{ai}, q_{bi}\right)\right)^2 \tag{10}$$

2.3 Test Data

For tests we have chosen motion of a left knee of healthy male, age 26. Recorded data samples count was equal to 812, but for experiments first 512 samples were used. Data were recorded with 100Hz frequency which gives recording duration approximately equal to 5 seconds - long enough to capture several steps. Figure 3 presents knee rotations in time as Euler angles for clarity. Angle values are truncated to the range of $\langle -180°; 180°)$. All results are also presented in this form. Time axis presents values always in seconds.

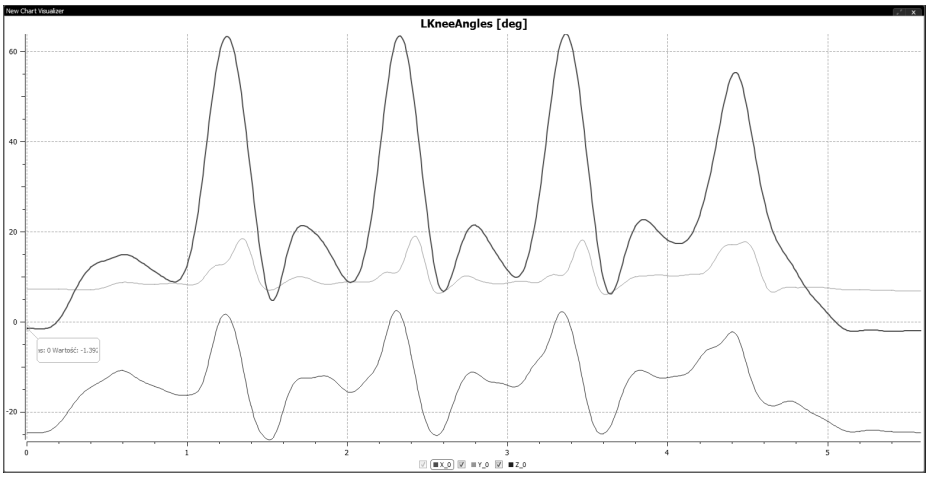


Fig. 3. Euler angles of test data - left knee of 26 years old and healthy man

2.4 Implementation

To verify presented lifting scheme we have implemented it within data flow processing framework for MDE software. The lifting scheme was decomposed to forward and inverse transforms, were each of them was modelled as an independent processing node. For testing a denoising process a dedicated source node was proposed allowing to set noise levels introduced to motion data (the σ value). For compression process two processing nodes were proposed - one for compression and other for decompression. Also signals differences was encapsulated in processing node. Based on such nodes several processing pipelines were created (Figure 4).

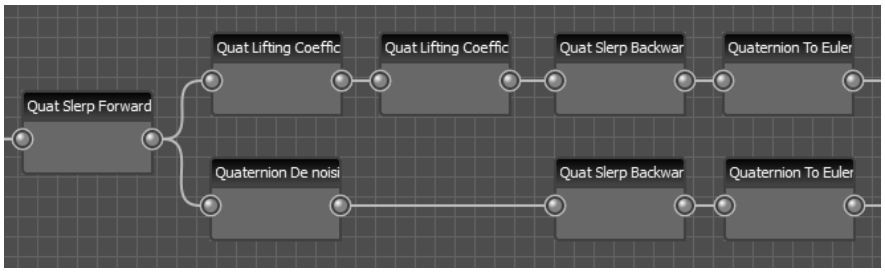
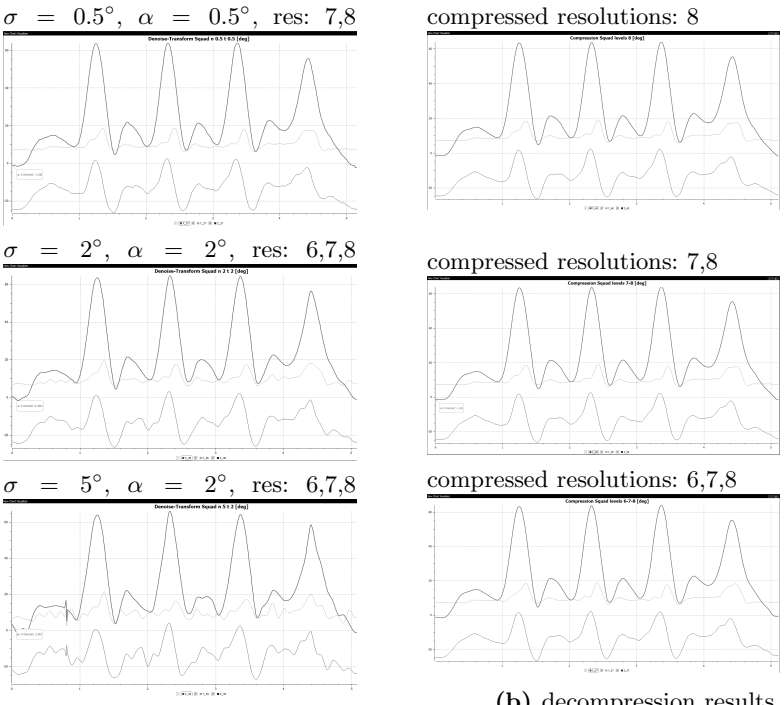


Fig. 4. MDE visual data flow environment for testing lifting scheme algorithms

2.5 Test Results

Noise Reduction. To test denoising, we have added three levels of white Gaussian noise for rotations in Euler angles representation, independently for each

angle, with σ equal to: 0.5, 2, 5 degrees. Next, the denoising was performed based on threshold α of the lifting scheme details coefficients from the selected resolutions. Visually results of denoising process are presented on Figure 5a. Distances of denoising signal after adding the noise to its original form are collected in Table 1, according to introduced quaternion signals distance measure.



(a) denoising results

(b) decompression results

Fig. 5. Processing based on squad lifting scheme

Compression. Testing compression abilities according to proposed compression method, three levels of compression were proposed, based on removed detail coefficients from resolution levels 6, 7, and 8 (Figure 5b). To compare decompression quality, the distance of reconstructed signal after decompression to original signal is measured (Table 2).

Table 1. Noise reduction results

Noise (σ[°])	Threshold angle (°)	De-noise details resolutions	Distance (radians)
0.5	0.5	7, 8	1.6
2	2	6, 7, 8	5.6
5	2	6, 7, 8	13.3

Table 2. Compression results

Levels of removed details resolution	Distance (radians)
8	0.006
7, 8	0.08
6, 7, 8	1.3

3 Summary

Proposed novel approach to motion data analysis with multiresolution tools for quaternions provide very promising results in the field of data compression and noise reduction. Presented technique opens new directions for motion analysis tools based on the quaternion lifting scheme. Test results were obtained with help of MDE software, proving that it can be easily adopted for custom research solutions, increasing their quality and standardising them for cooperation with other built-in functionality. This makes data analysis simpler and faster. Additionally, users can automatically utilise all available computational resources with help of delivered plug-in. The MDE software, through the idea of rapid prototyping, provides a new, very fast, and easy to use framework for testing various algorithms.

Presented motion data decomposition based on lifting scheme and proposed SQUAD prediction block can be used to create more reliable and descriptive motion processing tools, used further for motion classification, recognition and prediction.

Acknowledgment. This work was supported by projects NN 516475740 from the Polish National Science Centre and PBS I ID 178438 path A from the Polish National Centre for Research and Development.

References

1. Mokka Motion kinematic & kinetic analyzer, `http://b-tk.googlecode.com/svn/web/mokka/index.html`
2. Beaudoin, P., Poulin, P., van de Panne, M.: Adapting wavelet compression to human motion capture clips. In: Proceedings of Graphics Interface 2007, pp. 313–318. ACM (2007)
3. Fog, A.: The microarchitecture of intel, amd and via cpus. In: An Optimization Guide for Assembly Programmers and Compiler Makers. Copenhagen University College of Engineering (2011)
4. Hennessy, J.L., Patterson, D.A.: Computer Architecture: A Quantitative Approach. Elsevier (2012)
5. Hsieh, C.-C.: B-spline wavelet-based motion smoothing. Computers & Industrial Engineering 41(1), 59–76 (2001)
6. Hsieh, C.-C.: Motion smoothing using wavelets. Journal of Intelligent and Robotic Systems 35(2), 157–169 (2002)

7. Lee, J., Shin, S.Y.: A coordinate-invariant approach to multiresolution motion analysis. Graphical Models 63(2), 87–105 (2001)
8. Lee, J., Shin, S.Y.: General construction of time-domain filters for orientation data. IEEE Transactions on Visualization and Computer Graphics 8(2), 119–128 (2002)
9. Shen, J.P., Lipasti, M.H.: Modern processor design: fundamentals of superscalar processors, vol. 2. McGraw-Hill Higher Education (2005)
10. Shoemake, K.: Quaternion calculus and fast animation, siggraph course notes (1987)
11. Šilc, J., Robič, B., Ungerer, T.: Processor Architecture: From Dataflow to Superscalar and Beyond; with 34 Tables. Springer (1999)
12. Smith, J.E., Pleszkun, A.R.: Implementing precise interrupts in pipelined processors. IEEE Transactions on Computers 37(5), 562–573 (1988)
13. Szczesna, A., Slupik, J., Janiak, M.: Motion data denoising based on the quaternion lifting scheme multiresolution transform. Machine Graphics & Vision 20(3), 238–249 (2011)
14. Szczesna, A., Slupik, J., Janiak, M.: Quaternion lifting scheme for multi-resolution wavelet-based motion analysis. In: The Seventh International Conference on Systems, ICONS 2012, pp. 223–228 (2012)
15. Szczęsna, A., Słupik, J., Janiak, M.: The smooth quaternion lifting scheme transform for multi-resolution motion analysis. In: Bolc, L., Tadeusiewicz, R., Chmielewski, L.J., Wojciechowski, K. (eds.) ICCVG 2012. LNCS, vol. 7594, pp. 657–668. Springer, Heidelberg (2012)
16. Vicon Motion Systems. Polygon User Manual