I. Abstract

Real-time Kinematic data is a crucial piece of many emerging technology applications. These include virtual reality, autonomous driving, internet of things, physical therapy, wearable electronics, and human performance among others. These applications have been enabled by the recent explosion of cheap inertial based sensors (IMUs) along with mobile computation power to process this data at high sampling frequency. Kinematic data represents a nascent field of multimedia data which is starkly underdeveloped compared to the existing maturity of text, audio, and visual type data processing techniques. In order to enable the modern applications listed above, efficient and standard techniques for representing and processing kinematic data are needed.

This study explores a range of both traditional and novel compression techniques applied to human kinematic data gathered by 6-axis IMUs. The best performing techniques are identified and discussed.

This study focuses on highly practical approaches which can be used in modern applications. To demonstrate this, all compression techniques have been implemented and released as open-source C code using fixed point computations. The hope of the authors is that this code can be a starting point for academic or industry developers implementing some kinematic data application on any type of computational platform.

II. Introduction

- A. Significance of Kinematic data
- B. Significance of compression
- C. Relation to other multimedia
- D. Choice of data category and application
- E. Past work
- F. Hypothesis

III. Method

- A. Inclusion criteria for algorithms
  1. Lossless
  2. Low computation
  3. Causal
  4. Node independent
- B. Algorithms and formats
  1. Text
  2. Binary
  3. Rice encoding
  4. Difference encoding

5. Linear extrapolation
6. 2nd-5th order polynomial extrapolation
7. Natural spline
8. Rotating gravity
9. ML generated linear IIR filter

C. Data set description

IV. Results

FIG: Compression ratio of best performing configuration (filter order, Rice K, etc) of each algorithm, broken down by total, accelerometer, and gyroscope. (the point: which algorithm is best?)

FIG: High precision and fixed point comparison of ML FIR filters of increasing order (the point: how computationally complex can a filter be before numerical instability? Gives empirical, quantitative limits for our "low computation" inclusion factor.)

A. Compression levels of each algorithms
B. Comparison of Accelerometer and Gyroscope
C. Comparison of movement types or events
D. ML coefficients
E. Filter order and numerical stability

V. Discussion

A. Significant and easy gains can be made compared with traditional text
B. Surprising performance of difference encoding
C. Lack of correlation found between accelerator and gyroscope
D. Comparison to other compression fields
E. Future work

VI. Conclusion

A. Fixed point computation breaks down at around x computations
B. Use difference encoding
C. How this helps the field