Using Microsoft Kinect to Monitor Patients in the Home

Steve Sroba ssroba@andrew.cmu.edu
Heinz College
Carnegie Mellon University
Pittsburgh, PA, United States

Abstract

The main objectives for this analysis were to improve on past efforts of classifying physical activity recorded by Microsoft Kinect, to find new ways to analyze performance on common clinical assessments of elderly patients such as gait, balance, and strength, and to predict type of participant performing the movement. Classification of activity and type of participant were achieved using ensembles of classification trees trained on movement data collected by the Kinect, taken from the K3DA dataset. The classification accuracy achieved was 75.84% and 93.26% for activity and participant type, respectively. This is useful for identifying clinically relevant movements performed without explicit instruction (such as while playing Kinect games, or through background data collection) and for determining fitness level solely based on performing these movements, without any other information about the person performing the movement. Establishing new ways to analyze performance on common clinical assessments was not achieved, and thus is a future goal for this work.

1. Introduction

The Kinect is a 3D camera and microphone that can lead to change in health care such as increased virtual appointments, reduced transportation costs, service costs, and risk of infection. Daily assessments, physical therapy and exercise cam eliminate the need for frequent check-ups and office visits, and notify clinicians of declining health or risk of hospitalization. Games using Kinect can encourage active lifestyle or provide motivation for rehabilitation exercises(Phlmann et al. (2016)).

Around 1/3 of patients older than 65 and 1/2 older than 80 fall each year, and patients who have fallen, or have a gait or balance problem are at higher risk of falling again and losing their independence (Katherine T Ward and David B Reuben (2016)). On top of that, gait speed alone predicts functional decline and early mortality in older adults (Katherine T Ward and David B Reuben (2016)). For these reasons, assessing gait and balance may identify patients who need further evaluation.

Performing physical assessments can help clinicians manage conditions typical of aging adults and prevent or delay complications. Functional status can be assessed at three levels: basic activities of daily living (BADLs), instrumental/intermediate activities of daily living (IADLs), and advanced activities of daily living (AADLs). Scales that measure functional status at each of these levels have been developed and validated. The Vulnerable Elders Scale-13 (VES-13) is a 13-item screening tool that is based upon age, self-rated health, and the ability to perform functional and physical activities. The Katz index for ADLs and the Lawton scale for IADLs are also common indices. Some AADLs can be ascertained by using standardized instruments, but AADLs are more complex, and less elders can complete them. The NIH toolbox is a collection of assessments, accompanied by a study of performance by different demographics for comparison of individual performance. Meta-analyses have found

home assessments of this type to be effective in reducing functional decline as well as overall mortality (Katherine T Ward and David B Reuben (2016)).

Health care in the United States is extremely expensive for the quality of care that is given, and the elderly population accounts for a majority of health care spending. On top of that, the aging population is only going to make this crisis worse. Taking all of this into account, I believe this topic is extremely important analysis because it can lead to elimination of unnecessary office visits, prevention of adverse health events, and increased quality of care for patients, leading to higher reimbursements for physicians once quality based reimbursement is in effect.

The following sections include Background, model description, Experimental Setup, Results, Discussion and Related Work, and a Conclusion.

2. Background

Ensembles of decision trees were used in this study. Decision trees provide comprehensible models when trees are not too big, but usually are not the most accurate. Splits on nominal features have one branch per value, while splits on numeric features use a threshold. The key is that the simplest tree that classifies the training instances accurately will work well on previously unseen instances. Entropy is a measure of uncertainty associated with a random variable (Jeremy Weiss (2017)):

$$H(Y) = -\sum_{y \in \mathcal{Y}} p(y) log_2 p(y)$$

And conditional entropy is:

$$H(Y|X) = \sum_{x \in \mathcal{X}} p(X = x) H(Y|X = x)$$

where

$$H(Y|X = x) = \sum_{y \in \mathcal{Y}} p(Y = y|X = x) log_2 P(Y = y|X = x)$$

Decision trees use the concept of information gain, selecting the split S that most reduces the conditional entropy of Y for training set D (Jeremy Weiss (2017)):

$$InfoGain(D, S) = H_D(Y) H_D(Y|S).$$

These are the general steps for constructing a tree (Jeremy Weiss (2017)):

1. MakeSubtree(set of training instances D)
   (a) C = DetermineCandidateSplits(D)
   (b) if stopping criteria met
      i. make a leaf node N
      ii. determine class label/probabilities for N
   (c) else

i. make an internal node N
ii. S = FindBestSplit(D, C)
iii. for each outcome k of S
    A. Dk = subset of instances that have outcome k
    B. kth child of N = MakeSubtree(Dk)

(d) return subtree rooted at N

These are the general steps for determining splits for numeric features, which were used in this experiment. This should be run for each numeric feature at each node of decision tree induction (Jeremy Weiss (2017)):

1. DetermineCandidateNumericSplits(set of training instances D, feature Xi)
   (a) C = // initialize set of candidate splits for feature Xi
   (b) S = partition instances in D into sets s1...sV where the instances in each set have the same value for Xi
   (c) let vj denote the value of Xi for set sj
   (d) sort the sets in S using vj as the key for each sj
   (e) for each pair of adjacent sets sj, sj+1 in sorted S
      i. if sj and sj+1 contain a pair of instances with different class labels //assume using midpoints for splits
      ii. add candidate split Xi $\leq$ (vj + vj+1)/2 to C
   (f) return C

Decision tree depth is a parameter than can be optimized to best fit the data set. For example, setting the minimum leaf size tells the algorithm to stop if any split imposed on this node produces children with fewer than MinLeafSize observations. Setting the max number of splits tells the algorithm to stop when the algorithm splits MaxNumSplits nodes.

Ensembles are a set of learned models whose individual decisions are combined in some way to make predictions for new instances. To get diverse classifiers, bagging is used to choose different sub-samples of the training set, while boosting uses different probability distributions over the training instances. Random forests choose different feature sets and sub-samples. Bagging and random forests work mainly by reducing variance, while boosting works by primarily reducing bias in the early stages, and reducing variance in the latter stages (Jeremy Weiss (2017)).

Parameters can also be optimized for the ensemble. Number of trees is the number of unique decision trees to include in the ensemble. To train an ensemble using shrinkage, you can set the learning rate to a value less than 1, which requires more learning iterations, but often achieves better accuracy. Lastly, the number of variables to sample as well as the number of observations to observations to sample when building each tree can be modified.

Bayesian Optimization (Mockus (2013)) was used to optimize the parameters mentioned above. The prior captures our beliefs about the behaviour of the objective function, which was classification error in this case. After evaluation, the prior is updated to form the posterior distribution over the objective function. The posterior distribution is then used to construct an acquisition function that determines what the next query point should be, i.e. what parameters should be tested next.

3. Classification Tree Ensemble

This algorithm heavily relied on the Statistics and Machine Learning Toolbox and the Optimization Toolbox provided in Matlab. As described above, an ensemble of decision trees was chosen for classification by optimizing the hyperparameters associated with decision trees and ensembles. For both Exercises and Groups, a form of boosting called AdaBoostM2 was used. For the exercise model, 221 trees, 225 predictors, a learn rate of 0.7872, a minimum leaf size of 13, and a max number of splits of 110 was chosen. For the groups model, 15 trees, 1076 predictors, a minimum leaf size of 1, a max number of splits of 270, and a learn rate of 0.9444 was chosen.

	Exercises	Groups
METHOD	AdaBoostM2	AdaBoostM2
TREES	221	15
PREDICTORS	225	1076
LEARN RATE	0.7872	0.9444
MIN LEAF SIZE	13	1
MAX SPLITS	110	270

Code is available at `https://github.com/ssroba/ML-HC-final-project/`

4. Experimental Setup

Importing the data set and data wrangling was executed to get the data in the correct form to be analyzed. Signal processing and motion analysis was used for feature extraction/engineering. The MoCap toolbox was used for this, as well as to visualize the data (see figures in Appendix). The classifiers to consider were then narrowed down using the Classification Learner App in Matlab to get a general idea of which classifiers will perform the best. Comparing based on classification accuracy, the best classifiers were then trained and optimized for best performance. The optimization minimized the cross-validation loss (error) by varying the parameters. It found the hyperparameters that minimized five-fold cross-validation loss. During this process, feature selection was also performed by removing the least significant features after parameter optimization. Optimization was then repeated, and the entire process was repeated until the current model's classification error was higher than the previous model's.

4.1 Cohort Selection

The data set was created by Manchester Metropolitan University. It is titled K3Da, A Kinect Healthcare Dataset: Benchmarking Healthcare Applications. It is a realistic clinically relevant human action dataset containing skeleton and depth data and associated participant information. Activities are based on the Short Physical Performance Battery (Guralnik et al. (1994)), and were recorded in a lab-based indoor environment with a single Kinect One 3D sensor, fixed to a tripod and motions performed directly in front of the device.

The Population consists of 54 participants: 26 young ($\leq$ 59 years of age), 14 elderly ($\geq$ 60 years of age) and 14 British Masters Athletes.

	Young	British Master's Athletes	Old
AGE	≤ 59	≥ 60	≥ 60
NUMBER	26	14	14
DATA ID	1	2	3

The different activities performed are below.

Data ID	Exercise
01	Balance (2-leg Open Eyes)
02	Balance (2-leg Closed Eyes)
03	Chair Rise
04	Jump (Minimum)
05	Jump (Maximum)
06	One-Leg Balance (Closed Eyes)
07	One-Leg Balance (Open Eyes)
08	Semi-Tandem Balance
09	Tandem Balance
10	Walking (Towards the Kinect)
11	Walking (Away from the Kinect)
12	Timed-Up-and-Go
13	Hopping (one-leg)

4.2 Data Extraction

Each subject had a folder, and within each folder, every exercise they performed also has a folder containing CSV files of the X, Y, and Z coordinates for the 25 joints defined by the Kinect SDK, as well as the timestamps for all data points, and the timestamps for clean data points. The age, gender, weight, height and an indicator of the type of participant (1=young, 2=British Master's Athlete and 3=old) was provided for each in a separate CSV file.

The general process is outlined below.

1. extractData(K3Da)
    (a) kinectTable = table() // initialize data table
    (b) for patientFolder in K3Da
        i. for exerciseFolder in patientFolder
            A. if a raw data file exists: read both the raw and clean CSV file; extract the patient id, exercise id, trial id, and clean timestamps related to the raw file
            B. else continue to the next iteration of the loop
            C. if outlier file exists: read the timestamps; combine with the clean timestamps to get all timestamps for the raw file
            D. else: set timestamps = the timestamps from clean

E. if the number of timestamps and number of rows of raw data match: raw.time = timestamps; append data to kinectTable

F. else: continue to the next iteration of the loop

(c) return kinectTable

From there, the data was organized to make it easier to manipulate, such as creating a structure array with 1 row for each record (the data and time stamp fields still contained all data for the entire record). The mcfillgaps() function from the MoCap toolbox filled missing data, or data that was labeled as not "clean" in the original data set. Two participants were removed. One was because most of their data was missing, the other because they were the only participant with a "warning code" denoting that they did not finish the trial.

4.3 Feature Choices

The Matlab Signal Processing Toolbox and MoCap Toolbox were used to create the features. Standard deviation, mean, skewness, kurtosis, variance, max/mean/min velocity, max/mean/min acceleration, and total displacement of each joint in each axis was computed using the MoCap toolbox. The number of peaks, mean distance between each peak, and periodicity for each joint in each axis was computed using the Signal Processing Toolbox.

The data originally contained 1 x 75 vectors (3 axes, 25 joints) in each row/column intersection, so the next step was expanding the data so that every value had it's own cell and was an individual feature.

Out-of-bag predictor importance measure how influential the features are at predicting the response. The influence of a predictor increases with the value of this measure. If a predictor is influential, then permuting its values should affect the model error. This idea was used during training in order to narrow down the feature vector.

The method of feature selection embedded in model training is outlined below.

1. predictors = all
2. currentBestError = 0
3. overallBestError = 1
4. while currentBestError $\leq$ overallBestError
   (a) build and optimize ensemble with predictors
   (b) if currentBestError $\leq$ overallBestError
      i. overallBestError = currentBestError
   (c) find predictor importance
   (d) remove the predictors with the lowest 10% importance

4.4 Comparison Methods

There are trade-offs between several characteristics of algorithms, such as speed of training, memory usage, predictive accuracy on new data, and transparency or interpretability, meaning how easily you can understand the reasons an algorithm makes its predictions. The Matlab Classification Learner App was used to initially compare algorithms. The results are below.

Figure 1: Exercise Classification Learner Results

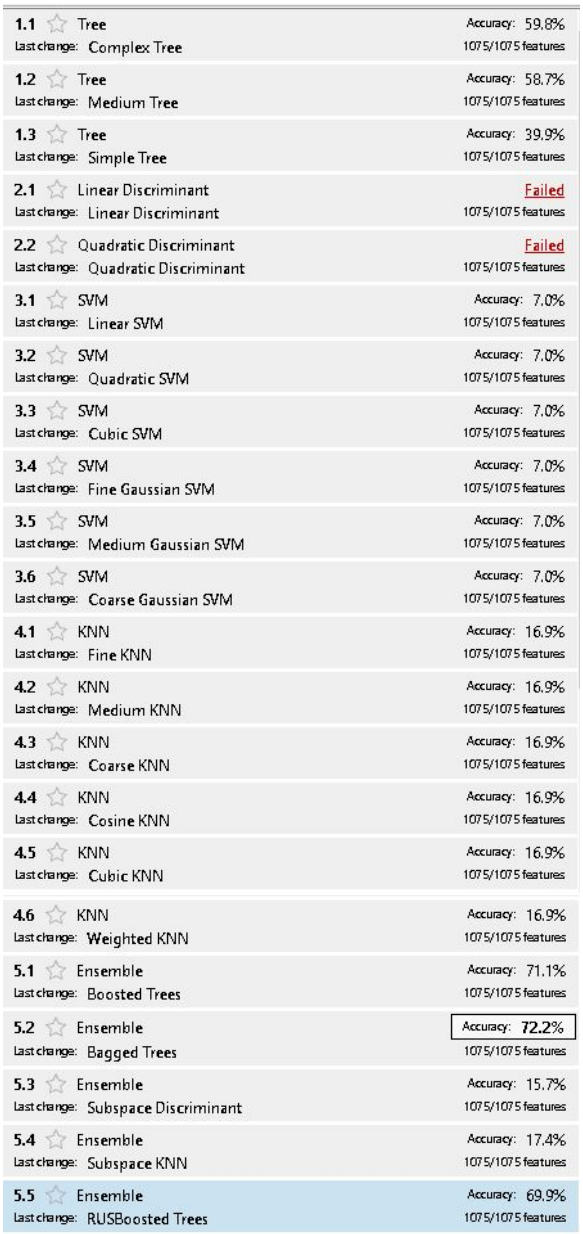
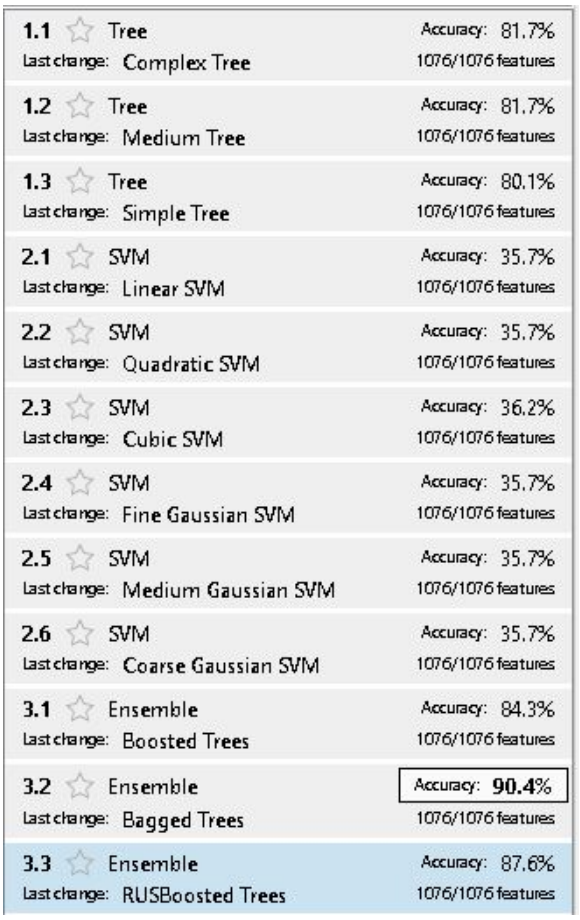


Figure 2: Group Classification Learner Results



From these results, decision tree ensembles were chosen for both groups and exercises. During training and parameter optimization, cross-validation error was used to compare models and return the best one.

4.5 Evaluation Criteria

An independent test set was left out of the training for evaluation. The models were trained on 2/3 of the data, and 1/3 was left for evaluation. Classification accuracy is adequate in this case because there is not a large class skew and there are not different misclassification costs. ROC and PR curves assume binary classification, so they were not considered, although could have been with more time. A confusion matrix was created to analyze which exercises/groups the models were mixing up. As expected, very similar movements such as balancing with eyes closed and eyes open were very hard to separate.

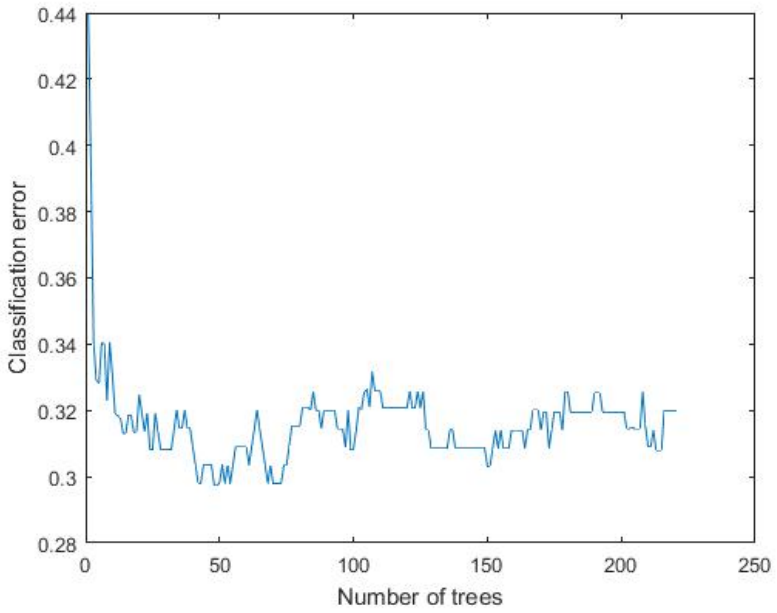
5. Results

	Exercises	Groups
CLASSIFICATION ACCURACY	75.84%	93.26%

5.1 Results on Exercises

The model classifying exercises achieved 75.84% classification accuracy on new data. The results as the number of trees increases are plotted below.

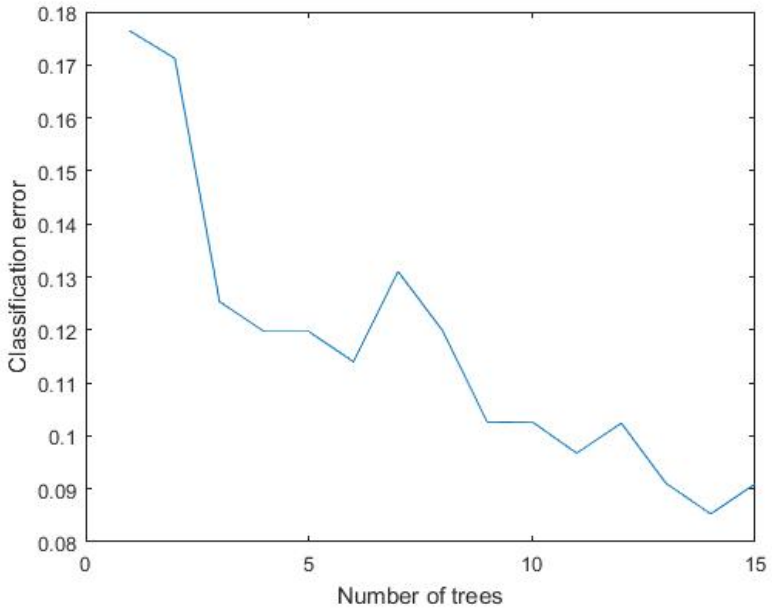
Figure 3: Exercise Classification Error



5.2 Results on Groups

The model classifying groups achieved 93.26% classification accuracy on new data. The results as the number of trees increases are plotted below.

Figure 4: Group Classification Error



6. Discussion and Related Work

Home healthcare can help elderly live an independent lifestyle in their own home and avoid costs of specialist facilities. Monitoring of normal activities, recognition of abnormal behavior, and detection of emergency situations can assure patient safety. Some examples that have been implemented are fall detection, as well as tremors and freezing in gait (Leightley (2015)). Also, the assessment of posture and body movement can provide important information for screening and rehabilitation. Some examples that have been implemented are detection of reduction in range of motion after surgery, gait and movement assessment for risk of falling and to predict patient's ability to cope with daily practice after discharge from hospital, and presenting rehabilitation exercises as a game to motivate patients to perform otherwise repetitive exercises (Alankus et al. (2010), Brennan et al. (2009), Rego et al. (2010)).

Leightley (2015) had many similarities, however this analysis attempted to improve upon it. Many more features were included in the classification. The comparison of classification accuracy is below.

	Exercises	Groups
THIS STUDY	75.84%	93.26%
Leightley (2015)	86.88%	NA

One limitation is that there are not enough subjects or data recorded in order to accurate results. Another is the credibility of the creation of the data set. Lastly, the data collection environment may have been too controlled for the results to be applicable in the real world.

7. Conclusion

Classification of exercises and types of participants was achieved using only Kinect movement data and decision tree ensembles. In the future, more advanced methods of feature representation (such as windowed signal analysis) and different models could improve upon these results. To add to this, I would also like to be able to combine the results with a scoring method to assess a patient's current status. Combining these classification tasks with scoring guidelines for clinical tests would make home-based assessment of elderly patients possible without any time or resources contributed by health care providers, leading to preventative care while decreasing the cost of care.

References

G. Alankus, A. Lazar, M. May, and C. Kellecher. Towards customizable games for stroke rehabilitation. In *SIGCHI Conference on Human Factors in Computing Systems*, page 2113 2122, 2010.

D. M. Brennan, S. Mawson, and S. Brownsell. Tele-rehabilitation: enabling the remote delivery of healthcare, rehabilitation and self management. In *Advanced Technologies in Rehabilitation volume 145*, 2009.

J. Guralnik, L. E. Simonsick, L. Ferrucci, R. Glynn, D. Blazer Berkman, P. Scherr, and R. Wallace. A short physical performance battery assessing lower extremity function: Association with self-reported disability and prediction of mortality and nursing home admission. In *Gerontology, 49(2)*, page 85 93, 1994.

Assistant Professor of Health Informatics Jeremy Weiss, MD-PhD. Decision trees and forests. In *Machine Learning for Health Care, Heinz College, Carnegie Mellon University*, 2017.

MD Katherine T Ward and MD David B Reuben. Comprehensive geriatric assessment. In *UpToDate, Inc.*, 2016.

D. Leightley. 3d human action recognition and motion analysis using selective representations. In *Faculty of Science and Engineering, School of Computing, Mathematics and Digital Technology, Manchester Metropolitan University.*, 2015.

Jonas Mockus. Bayesian approach to global optimization: theory and applications. In *Kluwer Academic*, 2013.

S.T.L. Phlmann, E.F. Harkness, and C.J. et al. Taylor. Evaluation of kinect 3d sensor for healthcare imaging. In *J. Med. Biol. Eng., 36*, page 857, 2016.

Paula Rego, Pedro Miguel Moreira, and Luis Paulo Reis. Serious games for rehabilitation: A survey and a classification towards a taxonomy. In *5th Iberian Conference Information Systems and Technologies (CISTI)*, page 1 6, 2010.

Appendix

Figure 5: Location Plot

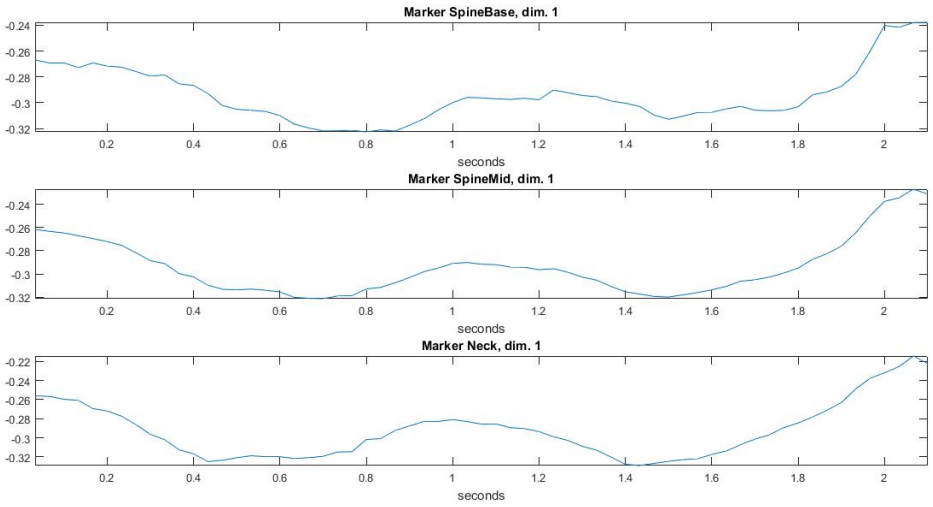


Figure 6: Acceleration Plot

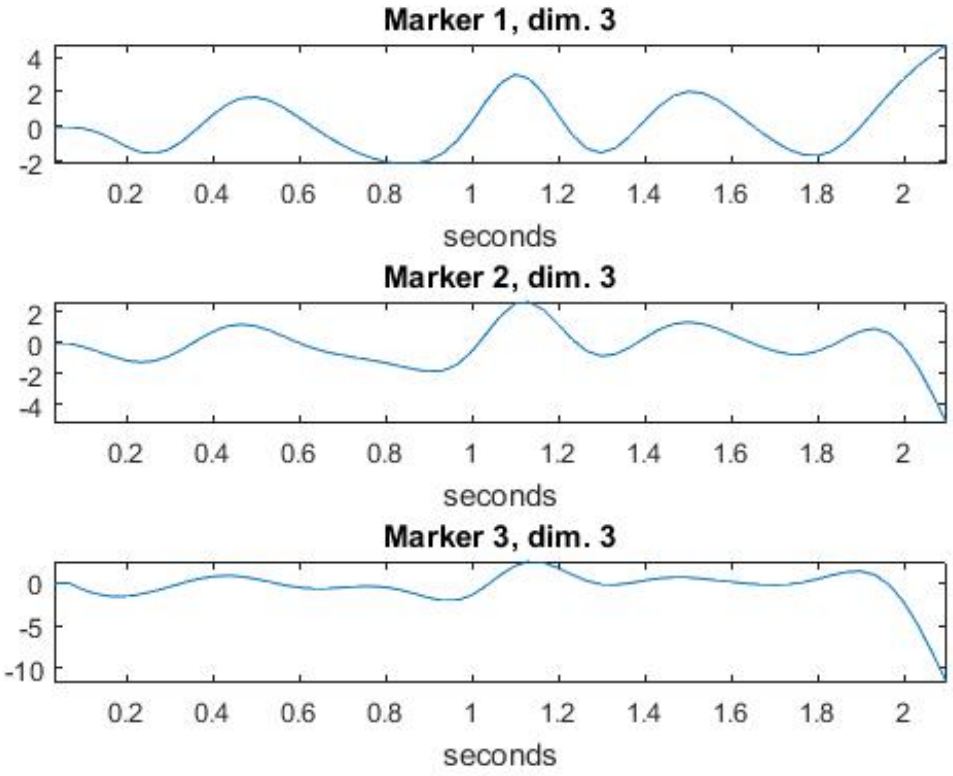


Figure 7: Velocity Plot