Automating Stroke Patient Evaluation using Sensor Data and SVM

(Invited Paper)

Paul Otten, Sang Hyuk Son, and Jonghyun Kim
Daegu Gyeongbuk Institute of Science and Technology
Email: {pcotten, son, jhkim}@dgist.ac.kr

***Abstract*—Evaluation of post-stroke hemiplegic patients is an important aspect of rehabilitation, especially for assessing improvement of a patient's condition from a treatment. It is also commonly used to evaluate stroke patients during theraputic clinical trials [1]. The Fugl-Meyer Assessment is one of the most widely recognized and utilized measures of body function impairment for post-stroke patients [2]. We propose a method for automating the upper-limb portion of the Fugl-Meyer Assessment by gathering data from sensors monitoring the patient. Features are extracted from the data and processed by a Support Vector Machine (SVM). The output from the SVM returns a value that can be used to score a patient's upper limb functionality. This system will enable automatic and inexpensive stroke patient evaluation that can save up to 30 minutes per patient for a doctor, providing a huge time-saving service for doctors and stroke researchers.**

I. Introduction

The Fugl-Meyer Assessment is a widely recognized method for performing a quantitative evaluation of a post-stroke patient's limb usage. It covers 5 domains: upper extremity motor, lower extremity motor, sensory, balance, and range of motion. The upper extremity motor section consists of 33 tests, which are movements performed by a patient (such as joint flexion or other limb movements). As the patient performs each movement, a score of 0 to 2 is given. The assigned scores generally follow these guidelines:

- 0: the patient cannot perform the movement at all
- 1: the patient can perform the movement partially
- 2: the patient can perform the movement faultlessly

The maximum score for the upper extremity motor is 66. The test typically takes a clinician about 30 minutes to perform for each patient.

We propose a system design that uses sensor data to automatically score a subject according to the Fugl-Meyer specification without the presence of a doctor or clinician. The subject simply sits in a chair in front of the Kinect sensor and a display with the user interface. The user interface instructs the subject to perform a series of movements, playing a video guide for each one. As the subject performs each movement, sensors observe the movement and pass the data to the main application to be processed. Once the application extracts useful features from the sensor data, it classifies the movement with an SVM, which gives a score of 0, 1, or 2 as its output. Our system supports 25 out of the 33 upper extremity motor tests, giving it a maximum possible score of 50. We believe that this provides a service to doctors treating stroke patients because it can save a significant amount of time for them. It also services researchers who are performing clinical trials on large numbers of stroke patients, since it gives an easy way for them to assess multiple stroke patients at periodic intervals with less effort than with currently available methods.

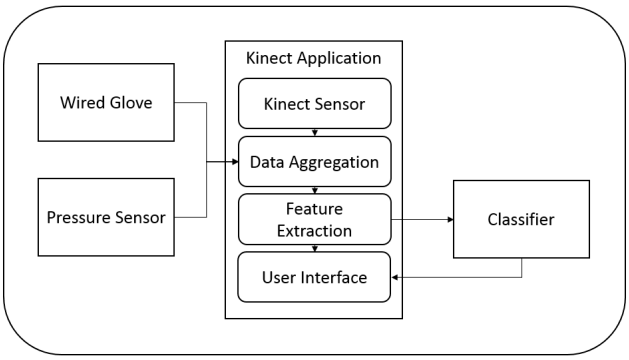
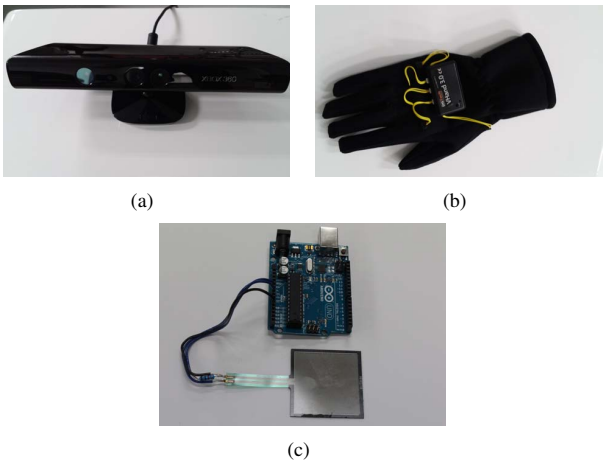


Fig. 1. An overview of our system design, showing the separate components and overall flow of operation.

The main sensor that the system uses is the Microsoft Kinect, which has an infrared camera that can be used with the Kinect SDK [3] to gather 3D skeleton information of up to two people in its view. The skeleton information partially consists of 3D coordinates for 16 pre-defined joints at 30 frames per second. The Kinect has already been used for similar gesture recognition applications [4], [5], [6] and can be used with additional sensors for our purposes.

In addition to the Microsoft Kinect, we used a pair of DG5-VHand Glove 3.0 sensors, originally developed for motion capture and gaming. This sensor is a glove that has flexion sensors sewn into the fingers. The glove also has an inertial measurement unit (IMU) with 9 degrees of freedom from its accelerometer, gyroscope, and magnetometer. This sensor is required because the Kinect skeleton information doesn't include any information about fingers or forearm pronation and supination. The VHand glove allows us to extend the number of upper extremity tests we can support. Our test coverage is further extended by the use of a simple pressure sensor. An overview of the sensors and system design are shown in Figure 1. The actual sensors we used in this design can be

978-1-4799-6833-6/14 $31.00 © 2014 IEEE
DOI 10.1109/SOCA.2014.29



(a) (b)

(c)

Fig. 2. The main sensors we used to record movement with our system. The Kinect in 2(a) is used for motion capture, the wired glove in 2(b) monitors the state of the hand and fingers, and the pressure sensor in 2(c) is used to measure grip strength.

seen in Figure 2.

With these sensors, we record approximately 5 seconds of the subject's movement. After we gather the sensor data, we run it through feature extraction routines. Each test has its own feature extraction routine because different movements are scored based on different factors. Once we extract our features, we train a SVM and use it to classify future movement recordings.

The main contribution of this paper is to provide a system design that can be implemented in a hospital to save time for doctors when they need to evaluate the limb functionality of a patient. Our system currently covers 76% of the Fugl-Meyer upper extremity motor tests and it can be extended to support additional tests with additional sensors. Even with partial test coverage, our system saves a significant amount of time for doctors who must evaluate multiple patients (approximately 30 minutes per patient). We believe that the use of this system will also encourage more frequent assessment, allowing patients to track their progress weekly or monthly. This is because patients may use this system as a tool to assess themselves without having to use up a clinician's time.

This paper is organized as follows: In Section 2, we discuss related works that use similar technologies to approach similar problems. In Section 3, we discuss the sensors that we use and how we collect data from them. In Section 4, we explain how we process the sensor data to find useful information that is used to distinguish between different levels of ability during movements. This section also includes information about how we set up the SVM classifiers. Section 5 outlines our experimental results and includes some benchmark information. The paper is concluded in Section 6.

II. Related Works

The core idea of our system is the combination of sensor data with machine learning to perform gesture recognition. Gesture recognition has been a widely studied subject with systems that approach the topic with various sensors and pattern recognition techniques. A very popular sensor used for gesture recognition is the IMU, which typically contains an accelerometer, gyroscope, and other sensors to measure motion or orientation.

There are several papers that demonstrated the feasability of using machine learning to classify gestures through features gathered from accelerometer data. In [7], the authors investigated different combinations of feature extraction methods to classify accelerometer data with SVM. In [8], the authors compared 7 feature extraction methods for classification with a nearest-neighbor classifier and found that the best feature sets achieved over 95% accuracy.

In papers such as [9], [10], the authors outlined methods for recognizing hand and pose gestures with computer vision. There have been a few vision based methods proposed, but this approach became more popular with the arrival of the Microsoft Kinect. Microsoft's Kinect SDK is a proven and versatile toolset that provides methods of easily obtaining skeletal information that can be used with machine learning algorithms such as SVM.

Some papers have investigated and demonstrated the feasibility of using machine learning to perform gesture recognition on Kinect data. Skeleton data representations were investigated in [11], [12] to allow for effective SVM feature extraction. The authors in [13] presented an approach similar to ours where Kinect skeleton data was processed and classified with multi-class SVM. They were able to distinguish between 8 different gestures with high accuracy. In [5], the authors used kinect data to score users' dance gestures by comparing them to a golden standard. The authors in [6] investigated the effectiveness of different machine learning algorithms for Kinect-based gesture recognition. The authors found that Backpropagation Neural Networks and SVM were much more accurate than Decision Tree learning or Naïve Bayes. In [4], the authors used SVM with another unsupervised clustering algorithm to classify aggressive actions with up to 90% accuracy.

There have also been medical rehabilitation applications of the Kinect and other sensor technology. Researchers at Microsoft are collaborating with Seoul National University to develop a physical therapy system with the Kinect[14]. Kinerehab is another physical rehabilitation system for motor disabilities [15]. However, these systems do not implement a method of providing standard evaluations. They are also limited to detecting limb positions without accurately knowing limb orientation. Therefore our system builds on many ideas presented in these works to provide an evaluation system that adheres to the Fugl-Meyer Assessment specification.

III. Sensor Data

The Fugl-Meyer Assessment requires a patient to perform movements as tests in front of a clinician. The clinician then scores the test based on how well the patient performs the movement. Our approach to automating the assessment is to replace the clinician with a set of sensors and a user interface

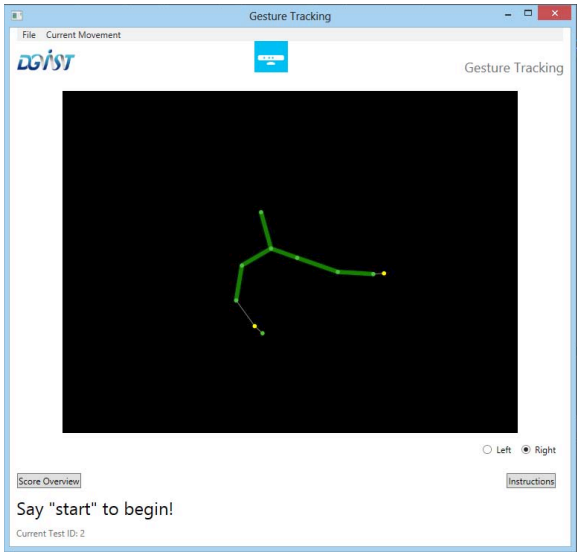


Fig. 3. The primary user interface for our system, which shows a graphical representation of the Kinect's sensor data.

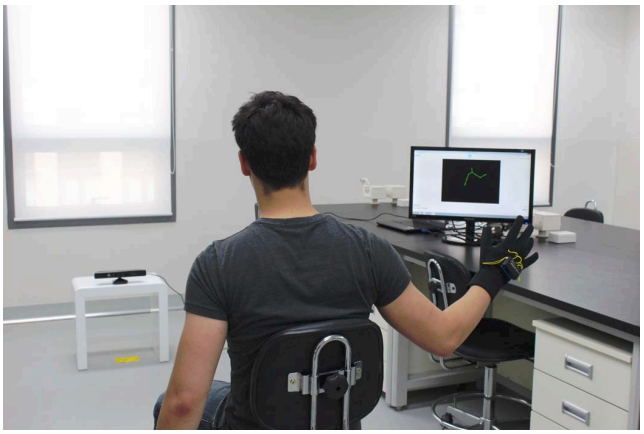


Fig. 4. The lab setup where the movement data from people performing the Fugl-Meyer Assessment is recorded.

as shown in Figure 3. The sensors that gather data are a Kinect, a wired data glove, and a separate pressure sensor. Figure 4 shows how the system is set up in our laboratory.

A. Kinect Data

The primary sensor used for the evaluation is the Kinect, which gets a depth image of the person performing the test movements. The Kinect SDK allows us to use this data to get the 3D positions of a person's head, center chest, shoulders, elbows, and wrists. These 3D joint positions are recorded at 30 frames per second as the subject performs each of their test movements.

The Kinect is directly integrated into the main application, which contains a buffer for Kinect skeleton frames. Once the user initiates sensor recording with a voice command, the system starts storing skeleton frames. After storing every 10th frame, the buffer is processed to check if a movement has occurred. This is done by checking the distances that the wrists have traveled three times a second, since the wrist is the furthest extremity we can reliably measure, and it moves a greater distance than the elbow or shoulder does during the tests. Once it detects motion and ensures that there hasn't been motion for another third of a second, it stops all sensor recording, gathers data from the other connected sensors, and starts feature extraction.

Despite being able to record the 3D positions of the user's joints, the Kinect has several limitations that need to be addressed to record movements such as forearm supination, pronation, and wrist dorsiflexion. The Kinect cannot perform reliable information about the orientation of the hand or fingers, which is required for a number of tests.

B. Wired Glove Data

The DG5-VHand Glove 3.0 is another sensor that we use to overcome the limitations of the Kinect. It is worn as a glove, and has flexion sensors in each of the fingers to detect how much each of the fingers can bend. There is also an IMU on the glove that is located on the back of the hand. The IMU contains an accelerometer, gyroscope, and magnetometer. This sensor allows us to support tests that would not be supported by the Kinect, such as tests requiring forearm supination and pronation. It also allows us to record movement data for tests already supported by the Kinect, such as speed or smoothness of the movement. This information is used to create extra features to aid classification of the movement. However, the main reason we use the glove is to support the ability to test finger movement and strength. The Fugl-Meyer test contains 5 different movements that test different grasps, along with an additional two movements that test finger range of motion.

C. Pressure Sensor

A third sensor that we use for two of our tests (grasp 2 and grasp 3) is a simple pressure sensor connected to an Arduino Uno. We used this sensor to measure the gripping strength between the index finger and thumb for these two tests. This sensor is required because the wired glove cannot detect any information about finger strength.

D. Data Aggregation

Each sensor is driven by an application that records the data from the sensors and sends them to the main application when requested. As the user selects a test movement to perform in the main application, he/she starts recording with a voice command. When the recording starts, the sensors start filling a buffer of sensor values. As data is being recorded, the main application analyzes the 3D joint position data to detect when the movement has stopped, at which point it gathers data from the other sensors and stops recording. If the user is unable to move during the recording window, the recording will time out after 5 seconds and gather the data at that time.

Communication between the main application, the sensors, and the classifier takes place through sockets, making the

components of this modular system easy to swap out. This has the benefit of making it easy to include additional sensors. To include another sensor, a new program is created for the new sensor following the template being used by the sensors that are already implemented. The program must be able to take commands from the main application, to record data, and to send the recorded data back to the main application. The main application can then simply add the additional sensor to the list of sensors it connects to. We have already demonstrated this extendability by adding support for the Shimmer3 sensor (a 9 degree-of-freedom IMU) and a custom data glove sensor based on the Arduino Uno.

IV. Machine Learning

In our system, SVM is the driving force behind the scoring of the subject's movements. Specifically, we used LIBSVM, a library for SVMs [16]. To use LIBSVM, training data with known classifications must be formatted into a text file and used to train a SVM classifier. After that, LIBSVM can classify further test instances with unknown classifications by comparing the test instance with the training instances and finding the best match. The main challenges with SVM are to decide what features to extract from the data and to establish the best parameters for training.

A. Feature Extraction

After receiving raw sensor data from the sensors, the system extracts usable data that can be used to differentiate between different levels of limb functionality. Since every test is a different movement, they have their own feature extraction routines and SVM classifiers. This is because each movement has different requirements for what should be observed to distinguish between levels of functionality. For example, wrist supination and pronation have to be measured by checking the range of wrist rotation while elbow flexion has to be measured by checking the angle of the elbow joint during the movement.

When the user performs a test and the main system collects the sensor data from the movement, it executes the feature extraction function for that test. Most of the tests need to process Kinect skeleton frames to generate features. Before they process the skeleton frames, they need to account for the variability in the user's movement, such as speed. We measure the speed of a movement by comparing the distance traveled by the wrist from frame to frame, since it is the furthest extremity that we can reliably measure with the Kinect. To account for speed variability and any pauses in the movements, the system uses a two-phase pre-processing routine.

During the first phase, the system compares the distance the wrist has traveled in adjacent frames, starting with the first two frames. For each pair of adjacent frames, if there is not a significant amount of movement, the second frame is removed. This has the effect of slightly speeding up the movement, and the speed difference is more noticeable for very slow movements. This step is required because many stroke patients often perform test movements with varying levels of speed throughout their full range of motion, sometimes including brief pauses along the way. After this first phase, the second phase is to set the number of remaining frames to a fixed number n. If the number of remaining frames is greater than n, then frames are removed at even intervals, having the effect of further speeding up the movement. If the number of remaining frames is less than n, then some frames are repeated at even intervals, having the effect of slowing the movement down. This pre-processing step is necessary to perform before feature extraction because each SVM classifier requires a fixed number of features, and the number of features the system creates is based on the number of skeleton frames. The fixed number n is different for some of the tests because we have found that some tests are classified more accurately with different values of n.

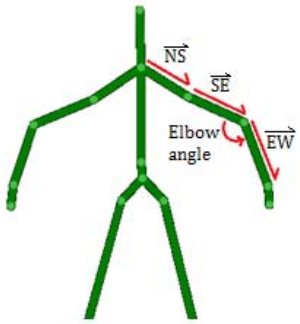
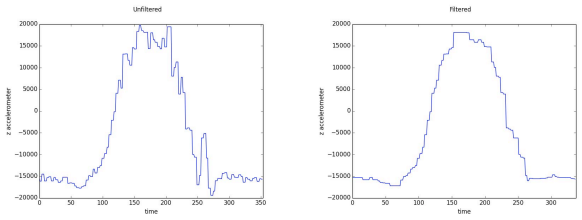


Fig. 5. A depiction of the skeleton data gathered by the Kinect with the features we extracted from that data. The limbs between the neck(N), the shoulder(S), the elbow(E), and wrist(W) are converted to 3D unit vectors. The elbow angle is calculated in degrees.

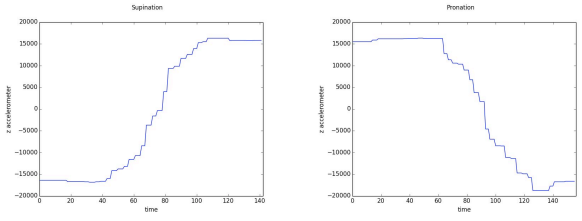
Once the feature extraction routine sets the number of skeleton frames to a fixed number, it begins gathering the actual features. The first set of features gathered is the direction that relative limbs are pointing over time. Limbs are composed of a superior joint j_s and an inferior joint j_i. The limbs relevant to the test movement are converted into 3D unit vectors, which gives information about the direction that the limb is pointing, but is not affected by the length of the limb. For every skeleton frame, the 3 elements of each unit vector for each limb are recorded as features.

The next set of features extracted from the skeleton frames is the set of angles between the relevant joints. For example, the Fugl-Meyer Assessment specifies that the elbow must remain at 90 degrees throughout certain movements. To calculate the angle a for a joint, we start with the unit vector for the superior limb $S =< s_0, s_1, s_2 >$ and for the inferior limb $I =< i_0, i_1, i_2 >$ that are adjacent to the joint. We then take the arccos of the dot product of these vectors and convert it to degrees, as shown in Equation 1. Each angle for each relevant joint in every skeleton frame is recorded as a feature. An illustration of the typical unit vectors and angles extracted as features is shown in Figure 5, where the angle for the elbow joint is calculated and a unit vector is created for 3 limbs: between the neck and shoulder, between the shoulder



(a) (b)

Fig. 6. The effect of median filtering on a window of accelerometer readings from a supination and pronation movement. Raw sensor data is shown in 6(a) while the smoothed version is shown in 6(b).



(a) (b)

Fig. 7. Z accelerometer sensor readings of 7(a) supination and 7(b) pronation.

and elbow, and between the elbow and wrist.

$$a = arccos[(s_0 * i_0) + (s_1 * i_1) + (s_2 * i_2)](\frac{180}{\pi}) \quad (1)$$

Many feature extraction routines require IMU data to overcome some of the shortcomings of the Kinect. For example, the Kinect is unable to accurately distinguish twisting motions such as supination and pronation. When working with the raw accelerometer and gyroscope data, the feature extraction functions use median filtering to smooth the data. This greatly simplifies feature extraction from the data, and the effect is shown in Figure 6.

For the tests that detect the range of supination or pronation, we use the sensor readings from the z accelerometer. The smoothed sensor readings for supination and pronation are illustrated in Figure 7. To determine the range of the motion, we identify windows in the sensor readings that look as they do in Figure 7 and use the difference between the maximum and minimum values. This approach is also used in other tests such as wrist circumduction and wrist dorsiflexion.

The grasp tests, which test a patient's finger functionality, are tested using the flexion sensors in the wired data glove. The values of the flexion sensors are higher the further the fingers are bent. Using this information, we take the readings from each finger and find the point where the average of all 5 finger readings are at their maximum in the set. That average is then used as the primary feature for these tests.

Some tests, such as wrist circumduction, use the "shakiness" of a movement as a factor for scoring. For these tests, we calculate a value representing the smoothness by using median filtering on the IMU data. We then sum the absolute value of the difference of each raw sensor value from the smoothed value. The resulting value is lower for smooth movements and higher for shaky movements, and is used as a feature for the tests that require it.

TABLE I
A COMPARISON OF SVM KERNEL FUNCTION ACCURACY ON OUR DATASET.

Linear	Polynomial	RBF	Sigmoid
83.33%	81.25%	33.33%	39.58%

B. Training

To train our classifiers, we gathered data from a non-stroke volunteer. This volunteer performed each test movement in a number of different possible ways. This gave us a dataset with multiple examples of each possible score for each supported test. Given this dataset, we made a few decisions about how we would train our classifiers.

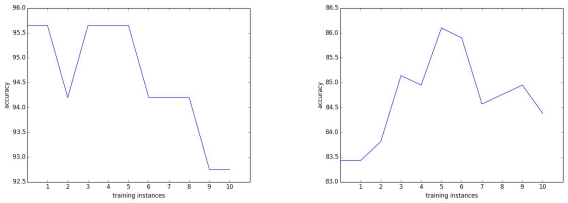
The first issue was to decide which SVM kernel function to use. The library we use, libsvm, provides 4 kernel functions: linear, polynomial, radial basis function (RBF), and sigmoid. We tested each of these kernel functions with a sample dataset and the default kernel parameters. The results, shown in Table I, indicate that linear kernels would be best suited for our system.

The linear kernel uses a C parameter to influence the size of the margin and location of the line separating the different training instances. We found that the highest accuracy is achieved when using a low value of C, setting it to 2^{-5}. Given this parameter, we trained each SVM classifier based on the data that we collected.

With these parameters, we ran tests to determine the best number of training examples to use. If the number of training instances were too small, the classifiers wouldn't have enough data to account for variations within the same possible classifications for each movement. If there were too many training instances, there may be an over-fitting problem where certain data points and features may have a negative effect on classification. This is counter-intuitive because logically, being shown more examples of a certain category should improve the classifier. However, this may cause the classifier to be affected by the values of certain features, resulting in a sub-optimal dividing line between data points from different classes.

To test the best number of training examples, we recorded a volunteer performing each test 30 times. For each test, the volunteer performed 10 instances of the movement in a way that would be classified as 0, 1, and 2 each. Afterwards, the same volunteer was recorded performing each test 3 times; once for each possible classification. This gave us an 11th set of data to be used for testing.

Figure 8(a) shows our results from testing the number of training sets incrementally. The accuracy shown is the overall accuracy for all tests combined. Most of the tests showed 100% accuracy regardless of the number of training examples. The variability in accuracy came from a small number of



(a) (b)

Fig. 8. The accuracy of our classifiers given the number of training examples for 8(a) the individual who created the training data and 8(b) the combination of data from all volunteers.

TABLE II
THE RESULTS OF CLASSIFICATION FROM OUR NON-STROKE VOLUNTEERS AFTER THEY PERFORMED EACH TEST.

Person	Accuracy	V-fold Accuracy
p_1	93.65%	90.48%
p_2	75.76%	86.36%
p_3	84.85%	89.39%
p_4	78.79%	84.85%
p_5	89.23%	90.77%
p_6	87.88%	87.88%
p_7	95.65%	89.39%
p_8	84.13%	96.83%

tests. When the number of training examples was between 3 and 5, there were only 3 tests that were not classified with 100% accuracy. When the number of training examples was 6 or greater, the number of tests that were not classified with 100% accuracy increased to 5. We repeated this test with all the data collected from volunteers with the results shown in Figure 8(b). These results indicate that 5 training sets for each class of each test is the optimal amount.

V. Experimental Results

The first step in testing our system was to assess the accuracy of our system using data collected from non-stroke volunteers. We gathered data from 8 volunteers, who were each asked to perform each test three times: once where the movement was performed faultlessly, once where it was performed partially, and once where it wasn't performed at all. This gave us a data set of test movements resulting in scores of 0, 1, and 2. With this data set, we gathered information for our feature extraction routines and the amount of training data required for our system to provide reliable results.

After determining the best number of training instances as discussed in the previous section, we tested the system using data from several volunteers. To get another measure of accuracy, we also performed v-fold cross validation on the data collected from our volunteers. The results, shown in Table II, indicate that our system is capable of maximum accuracy of up to 96.83%, with an overall average accuracy of about 86.1%.

The main reason for the inaccuracies in testing our sensor data is the noise from the Kinect sensor. When initializing the Kinect, the system enables the skeleton joint coordinate smoothing features implemented in the Kinect SDK. Despite this, there is often a lot of jittering with many of the joints when recording with the Kinect. In a number of cases, the Kinect miscalculates the position of an entire arm, especially if there is any occlusion. To account for this, we included noisy data into our training sets. When there is severe occlusion, the volunteers are given instructions to adjust the direction they were facing relative to the Kinect.

Our system was tested on a laptop with an Intel Core i7 processor and 8 GB of RAM. Given these specifications, it typically took our system about 1.5 seconds to extract features from sensor data and classify them for each movement. Therefore, our system is capable of recording a movement and classifying it in real time.

VI. Conclusion

In this paper, we proposed a method for using sensors to automate the Fugl-Meyer Assessment to evaluate the upper-limb condition of post-stroke patients. We have implemented the system as a proof-of-concept to demonstrate feasibility and are in the process of receiving approval for actual stroke patient tests. Once the reliability of this system is demonstrated, it will offer an inexpensive method for automating post-stroke patient upper limb evaluation. The primary benefit of this system is that it provides a method of evaluation that can be performed remotely without the presence of a doctor. This system saves up to 30 minutes of a doctor's time for each patient that they have, providing an inexpensive, time-saving service. Our preliminary experiments on healthy volunteers have shown that our system can achieve an overall accuracy of 86.1%.

Given our results, we envision a few different ways to extend the work. With the design we created, the application presented in this paper can be extended to support more tests using more sensors. To add support for an additional test, an additional feature extraction routine should be created that specifies relevant information to the test. This routine has to specify features to be gathered from the available sensor data. Typical features include elbow and shoulder joint angles, forearm and upper arm limb directions, and hand orientation.

In addition, accuracy of the tests currently supported can be increased by specifying different features, utilizing data from an additional sensor, or using another machine learning algorithm for classification. In [6], Backpropagation Neural Networks were shown to be a promising candidate for classification of Kinect data.

To test the usefulness of our application, tests should be performed with actual stroke patients. We will use stroke patient data to train our classifiers to ensure that we record the typical tendencies of stroke patients as they perform the test movements to improve our accuracy. We are currently in the process of requesting approval to record this data and perform tests with actual patients. Once we receive the approval, we will repeat the tests that we performed on healthy volunteers to find the accuracy of our system in a realistic setting. We will also have the stroke patient movements scored by clinicians to compare those scores with the scores given by our system.

ACKNOWLEDGEMENTS

This research was supported in part by the DGIST Research and Development Program of the Ministry of Science, ICT and Future Planning of Korea (CPS Global Center).

REFERENCES

[1] K. J. Sullivan, J. K. Tilson, S. Y. Cen, D. K. Rose, J. Hershberg, A. Correa, J. Gallichio, M. McLeod, C. Moore, S. S. Wu *et al.*, "Fugl-meyer assessment of sensorimotor function after stroke standardized training procedure for clinical practice and clinical trials," *Stroke*, vol. 42, no. 2, pp. 427–432, 2011.

[2] A. R. Fugl-Meyer, L. Jääskö, I. Leyman, S. Olsson, and S. Steglind, "The post-stroke hemiplegic patient. 1. a method for evaluation of physical performance." *Scandinavian journal of rehabilitation medicine*, vol. 7, no. 1, pp. 13–31, 1974.

[3] Kinect sdk. [Online]. Available: http://www.microsoft.com/en-us/kinectforwindows

[4] S. Nirjon, C. Greenwood, C. Torres, S. Zhou, J. A. Stankovic, H. J. Yoon, H.-K. Ra, C. Basaran, T. Park, and S. H. Son, "Kintense: A robust, accurate, real-time and evolving system for detecting aggressive actions from streaming 3d skeleton data," in *Pervasive Computing and Communications (PerCom), 2014 IEEE International Conference on*. IEEE, 2014, pp. 2–10.

[5] D. S. Alexiadis, P. Kelly, P. Daras, N. E. O'Connor, T. Boubekeur, and M. B. Moussa, "Evaluating a dancer's performance using kinect-based skeleton tracking," in *Proceedings of the 19th ACM international conference on Multimedia*. ACM, 2011, pp. 659–662.

[6] O. Patsadu, C. Nukoolkit, and B. Watanapa, "Human gesture recognition using kinect camera," in *Computer Science and Software Engineering (JCSSE), 2012 International Joint Conference on*. IEEE, 2012, pp. 28–32.

[7] Z. He, "Accelerometer based gesture recognition using fusion features and svm," *Journal of Software*, vol. 6, no. 6, pp. 1042–1049, 2011.

[8] S. J. Preece, J. Y. Goulermas, L. P. Kenney, and D. Howard, "A comparison of feature extraction methods for the classification of dynamic activities from accelerometer data," *Biomedical Engineering, IEEE Transactions on*, vol. 56, no. 3, pp. 871–879, 2009.

[9] Y. Wu and T. S. Huang, "Vision-based gesture recognition: A review," in *Gesture-based communication in human-computer interaction*. Springer, 1999, pp. 103–115.

[10] A. Erol, G. Bebis, M. Nicolescu, R. D. Boyle, and X. Twombly, "Vision-based hand pose estimation: A review," *Computer Vision and Image Understanding*, vol. 108, no. 1, pp. 52–73, 2007.

[11] L. Xia, C.-C. Chen, and J. Aggarwal, "View invariant human action recognition using histograms of 3d joints," in *Computer Vision and Pattern Recognition Workshops (CVPRW), 2012 IEEE Computer Society Conference on*. IEEE, 2012, pp. 20–27.

[12] M. Raptis, D. Kirovski, and H. Hoppe, "Real-time classification of dance gestures from skeleton animation," in *Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. ACM, 2011, pp. 147–156.

[13] K. Biswas and S. K. Basu, "Gesture recognition using microsoft kinect®," in *Automation, Robotics and Applications (ICARA), 2011 5th International Conference on*. IEEE, 2011, pp. 100–103.

[14] M. Lee, "Stroke recovery gets a boost from kinect," blog, Jan 2014. [Online]. Available: http://blogs.msdn.com/b/msr_er/archive/2014/01/22/stroke-recovery-gets-a-boost-from-kinect.aspx

[15] J.-D. Huang, "Kinerehab: A kinect-based system for physical rehabilitation: A pilot study for young adults with motor disabilities," in *The proceedings of the 13th international ACM SIGACCESS conference on Computers and accessibility*. ACM, 2011, pp. 319–320.

[16] C.-C. Chang and C.-J. Lin, "Libsvm: a library for support vector machines," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 2, no. 3, p. 27, 2011.