# Kintense: A Robust, Accurate, Real-Time and Evolving System for Detecting Aggressive Actions from Streaming 3D Skeleton Data

Shahriar Nirjon, Chris Greenwood, Carlos Torres, Stefanie Zhou, and John A. Stankovic
University of Virginia
{smn8z,cmg7t,ct5ab,xz5xm,jas9f}@virginia.edu

Hee Jung Yoon, Ho-Kyeong Ra, Can Basaran, Taejoon Park, and Sang H. Son
Daegu Gyeongbuk Institute of Science and Technology
{heejung8, hk, cbasaran, tjpark, son}@dgist.ac.kr

***Abstract*—Kintense is a robust, accurate, real-time, and evolving system for detecting aggressive actions such as hitting, kicking, pushing, and throwing from streaming 3D skeleton joint coordinates obtained from Kinect sensors. Kintense uses a combination of: (1) an array of supervised learners to recognize a predefined set of aggressive actions, (2) an unsupervised learner to discover new aggressive actions or refine existing actions, and (3) human feedback to reduce false alarms and to label potential aggressive actions. This paper describes the design and implementation of Kintense and provides empirical evidence that the system is $11\% - 16\%$ more accurate and $10\% - 54\%$ more robust to changes in distance, body orientation, speed, and person when compared to standard techniques such as dynamic time warping (DTW) and posture based gesture recognizers. We deploy Kintense in two multi-person households and demonstrate how it evolves to discover and learn unseen actions, achieves up to $90\%$ accuracy, runs in real-time, and reduces false alarms with up to $13$ times fewer user interactions than a typical system.***

## I. Introduction

Studies show that $30\% - 50\%$ of the patients with cognitive disorders suffer from various forms of agitation [9], [22]. Three major factors [22] are used to describe agitation among the demented elderly: verbal aggression, physical aggression, and antisocial behavior. Of these, physical aggression is of our interest as statistics indicate that $11\%$ of the hospitalizations in persons with cognitive disorders are due to physical aggression [31]. While most assisted living facilities rely upon caregivers to monitor their patients, the job of a caregiver can be psychologically demanding, which may result in psychiatric symptoms of caregiver burnout [25]. Hence, an automated system that is capable of continuously monitoring agitation is gaining popularity.

State-of-the-art agitation monitoring systems use two kinds of sensing techniques in general: invasive [26], [15], [6] and non-invasive [1], [12], [21]. Invasive techniques use physiological parameters such as EEG waveforms, skin temperature, skin conductance, pupil diameter, respiration rate, and accelerometer readings from wearable sensors to detect agitation. These systems require special devices to be worn at all times and are not convenient for a long term monitoring. Non-invasive techniques, on the other hand, mostly use movement sensors such as a bed exit sensor, door reed switches, and surveillance cameras. While these may be practical for long-term monitoring, they do not provide information at the skeleton level which could be analyzed to detect aggressive moves by the patient. Kinect [2] sensors are an exception which provide skeleton joint level information using a non-invasive infrared camera. Being inspired by its potential, in this paper, we address the problem of detecting and discovering aggressive actions using Kinect.

Kinect has successfully been used in solving problems such as gesture recognition [17], [19], [5], [24], [33], [27], fall detection [28], and gait analysis [13], [29]. To the best of our knowledge, we are the first to detect and discover multiple aggressive actions in real-time and in a real-world setup. Detecting aggressive behaviors with Kinect is challenging for several practical reasons. First, different people have different ways of expressing the same aggressive action. Training a system that works for everyone is difficult because of the variety in pose, velocity of movement, and body structure. Second, there are issues in a real-world setup, such as the distance and orientation of human body with respect to the Kinect, which need to be taken care of – as the device is meant to be used for playing video games where the player always faces the camera and stands at a recommended distance. Third, the system should have a way of detecting potential aggressive actions that are not in its current library of aggressive actions and add them into the library for taking them into account in future. Fourth, false alarms are very common in any safety system. Handling false alarms properly so that the system reduces them over time is important.

In this paper, we present Kintense which is a real-time, accurate, robust, and evolving system for detecting aggressive actions in a home environment using 3D skeleton data from Kinect. Kintense uses a combination of supervised and unsupervised machine learning techniques to recognize aggressive actions such as hitting, kicking, pushing, and throwing that are in its current library of aggressive actions, and to identify potential aggressive actions which a human user may label and add to the library for future detections. Unlike other existing Kinect-based gesture recognition systems, Kintense is robust to changes in relative distance between the body and the sensor, skeleton orientation, and speed of an action. Kintense is an evolving system which takes user feedback to learn new actions and new examples of an existing action, and to reduce false alarms to improve its accuracy.

We evaluate Kintense with both controlled and uncontrolled experiments. To evaluate the action recognition algorithm, we perform a series of experiments using our empirical dataset, where we compare the accuracy, classification delay, and robustness of Kintense with a dynamic time warping (DTW) based gesture matching algorithm and a posture template based gesture recognizer. To show that Kintense works in a real-world setup, we perform uncontrolled experiments in two multi-person households and demonstrate how Kintense detects the actions of interest – accurately and in real-time, as well as how it discovers new actions and evolves itself to learn and recognize them.

The contributions of this paper are the following:

- A gesture dataset [4] recorded with Kinect that consists of time-stamped skeleton data from 19 people, which includes 10 actions, about $13,000$ instances, and $7.4$ hours of data. To the best of our knowledge, this is the largest publicly available dataset that contains so many variations in user locations and body-orientations.
- An array of supervised classifiers, each of which recognizes a specific aggressive action by computing: distance, body-orientation, person, and time invariant skeletal features from streaming 3D skeleton data. These classifiers are empirically shown to be $11\% - 16\%$ more accurate and $10\% - 54\%$ more robust when compared to two standard techniques.
- A semi-supervised classifier that discovers unseen and potentially aggressive actions, and uses hierarchical clustering to get them labeled by a human operator with less user interactions.
- Kintense, a complete system that discovers and recognizes aggressive actions using a combination of supervised and semi-supervised classifiers. Two deployments of Kintense in two multi-person households demonstrate how the system evolves to discover and learn unseen actions, achieves up to $90\%$ accuracy, runs in real-time, and reduces false alarms with up to 13 times fewer user interactions than a typical system.

## II. Challenges and Design Goals

Fundamentally, detecting aggressive actions from a sequence of *stick figures* should be no different than detecting regular actions. However, practical issues involving the human subjects being monitored, sensing in a real-world environment, and some requirements specific to the problem domain – altogether make it challenging.

### A. Diversity in People and Actions

There is diversity in people and the way they exhibit an aggressive action. People vary in height, body structure, and pace. The speed of an action also varies from one instance to another, even if it is performed by the same individual. Furthermore, there is heterogeneity in styles and poses of an action. It is highly likely that, when an automated monitoring system is deployed, new ways of performing an aggressive action will be exhibited by the person under monitoring. Hence, an automated detector should be capable of absorbing these diversities in people as well as the diversities in their actions.

### B. Sensing in a Real-World Environment

Sensing in a real-world environment brings new challenges in addition to the core action recognition problem. In a real-world scenario, the person being monitored is not always facing the camera as he is in a game playing scenario; rather he is engaged in his daily activities and sometimes roaming around the room. Therefore, care must be taken to handle the relative distance and body orientation between the person and the Kinect. Occlusion of body joints is another common phenomenon. Sensor readings obtained from Kinect in such cases are extremely noisy and properly compensating for noise is mandatory for an accurate classification of actions.

TABLE I. Assessment Scales and Aggressive Actions.

| Scale | Physically Aggressive Actions |
|---|---|
| CMAI [9] | bite, grab, hit, kick, push, scratch, throw, wander. |
| RAGE [22] | bite, hit, kick, pinch, push, scratch, shove, throw. |
| BARS [11] | grab, hit, pace, push, restlessness. |

### C. Extensibility

Historically there have been several scales for assessing cognitive disorders [9], [22], [11]. Each of these scales describes a different set of actions as physically aggressive one as shown in Table I. While some actions such as hitting, kicking, and pushing are common in them, many are not, and there are at least a dozen such scales used by the practitioners [14]. We find it infeasible to train a system for every possible aggressive action. Instead, we create an array of expert classifiers which are trained for recognizing a predefined set of actions, and set extensibility as one of our design goals which allows the system to grow over time by automatically discovering potential aggressive actions and learning them to recognize their future occurrences.

### D. False Alarms

False alarms are common in any safety system. However, an advanced system should be able to reduce the amount of false alarms over time. To facilitate this, we employ the notion of *people in the loop*. We believe that an automated system, especially which relates to human safety, should not be left totally on the judgment of the machine, while at the same time it should not bug the human operator at every single occurrence. A balance between the two is to be made so that the system learns and reduces false alarms over time, while lessening the burden on the operator of correcting the system as much as possible.

## III. Overview of Kintense

Kintense is composed of three major architectural components: sensing, classification, and action discovery and refinement via user feedback. This section briefly describes these components in order to provide a high-level overview of the system. The algorithmic details of Kintense are described in the next two sections.

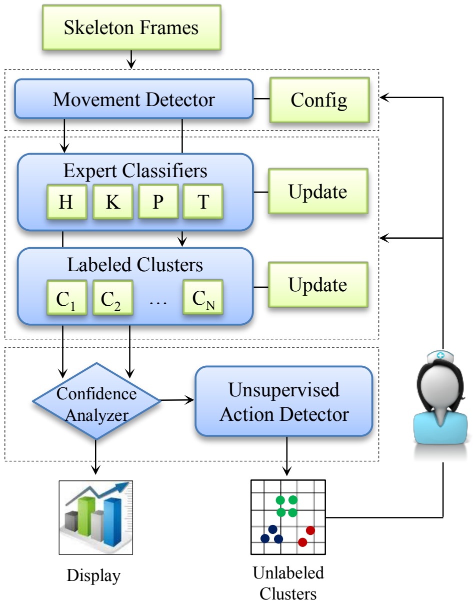

Fig. 1. The system architecture of Kintense comprising of expert classifiers, cluster-based classifiers, and user feedback.

## A. Sensing and Movement Detection

Kintense uses the streaming skeleton data from the Kinect to detect and classify actions. Kinect recognizes up to 6 users within its field of view, and by default, selects the first two users to track 20 joints on each of their skeletons. Skeleton frames are generated at the rate of 30 frames per second, and each frame consists of the 3D coordinates of 20 body joints along with their tracking states (tracked, inferred, or not tracked). The 3D coordinates are with respect to a frame of reference centered at Kinect.

Kintense accumulates a window of $w$ frames and analyzes it to detect skeletal movements. The `movement detector` is a simple threshold-based classifier which acts as an admission controller for the classification phase. It computes the variances of joint coordinates, compares them against a set of predefined and configurable thresholds, and if any of the values are above some threshold, the window is passed on to the classifiers.

## B. Action Classification

Kintense employs an array of binary classifiers, each of which recognizes a specific aggressive action. The classifiers are of two types: *expert classifiers* and *labeled cluster-based classifier*. Both of these are similar in the sense that they classify a window of frames and provide a confidence value of that window's likelihood of belonging to a specific action class. However, their differences are: the expert classifiers are offline-trained, aggressively tested, highly tuned to recognize specific actions, fixed in number, and provide better classification results in general. The labeled clusters, on the other hand, are obtained by unsupervised learning and are labeled by a human operator. These clusters are used to represent different actions and are used by a Bayesian classifier to recognize them. As the system runs and discovers more action instances, new clusters are added, old ones are updated, and the accuracy of the cluster-based classifier is improved.

The rationale behind having two types of classifiers is to achieve a high recognition accuracy for actions that are common in most agitation scales as well as to be able to learn actions that are new or vaguely defined. There is no fixed list of aggressive actions with precise definitions that we can follow to design expert classifiers for all actions. On the other hand, evolving a classifier as done in the labeled cluster-based method requires comparatively longer time to attain an acceptable recognition accuracy.

## C. Action Discovery and Refinement

An action that is classified poorly by both types of classifiers corresponds to either a new action that was never seen before or a different way of performing an existing action. In both cases, frames containing such actions are handed over to the `action detector` module. At this stage, at first, a snapshot of the event, consisting of time-stamped skeleton frames and a small thumbnail sized image, is created. Then an extended feature vector comprising of a sequence of poses are computed and stored in a database along with the snapshot. When the number of such examples grows beyond a certain amount, they are clustered, ranked, and presented before a human operator for tagging with appropriate labels.

The rationale behind choosing this approach of clustering potential actions is to lessen the burden on human operator in tagging actions. Clustering helps reduce the amount of supervision needed from the operator and snapshots provide visual assistance in tagging an action with ease. Ranking of clusters provides a way of filtering out unnecessary tagging operations. For example, in our experience, the more frequent clusters often represent common actions in daily activities which constitutes most of the false alarms.

## IV. Expert Classifiers

Kintense has four supervised classifiers for recognizing four common aggressive actions: hitting, kicking, pushing, and throwing. Each of these are binary classifiers recognizing an action from a window of skeleton frames (or stick figures) obtained from Kinect. Frames from Kinect are first converted into feature vectors which are invariant to relative position and orientation of the body, and speed of an action. The feature vectors are used to train support vector machine (SVM) classifiers. This section describes how the feature vectors are computed and the classifiers are created.

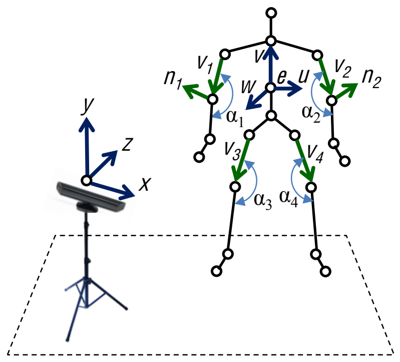

Fig. 2. An illustration of *body relative features* that are used in Kintense.

## A. Body Relative Coordinates

The design of feature vectors in Kintense is inspired by our natural way of describing an action. Actions are best described by the relative movements of our body parts with respect the torso. Hence, at first, we establish a body relative frame of reference and use it to redefine the 3D coordinates of all 20 skeleton joints that we get from Kinect. The new frame of reference is described by the origin $\mathbf{e}$ and three basis vectors $\{\mathbf{u}, \mathbf{v}, \mathbf{w}\}$, which are shown in Figure 2. The origin $\mathbf{e}$ is located at the spine joint, $\mathbf{u}$ points toward the direction from the right shoulder to the left, $\mathbf{v}$ points upward along the spinal cord, and $\mathbf{w}$ is normal to both $\mathbf{u}$ and $\mathbf{v}$ and points forward. The transformation of coordinates of a point in Kinect's coordinate system $\mathbf{p}_{xyz}$ to the body relative coordinate system $\mathbf{p}_{uvw}$ is computed by the following formula:

$$\mathbf{p}_{uvw} = \begin{bmatrix} \mathbf{x}_{uvw} & \mathbf{y}_{uvw} & \mathbf{z}_{uvw} & \mathbf{o}_{uvw} \\ 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{p}_{xyz} \qquad (1)$$

where $\mathbf{x}_{uvw}$, $\mathbf{y}_{uvw}$, $\mathbf{z}_{uvw}$, and $\mathbf{o}_{uvw}$ are the $(u, v, w)$ coordinates of Kinect's $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{z}$ axis and origin $\mathbf{o}$.

## B. Pose Modeling

An action is a sequence of poses over time. Hence, prior to modeling an action, we model the poses in terms of relative positions and orientations of body parts. The four actions that we consider require modeling only the arms and the legs, each of which have two large bones that are modeled as follows.

*1) Modeling Upper Arm and Thigh:* These bones are attached to the torso at the ball-and-socket joints and move freely in 3D. These four bones are modeled as four 3D unit vectors $v_1, v_2, v_3$, and $v_4$ as shown in Figure 2, and are computed from the coordinates of the endpoints.

*2) Modeling Lower Arm and Leg:* These bones only bend $0° - 180°$ at the elbow and knee joints. We model their relative positions with respect to the upper bones using four angles $\alpha_1, \alpha_2, \alpha_3$, and $\alpha_4$ as show in Figure 2. Besides these angles, we also keep track of the planes containing the upper and lower arms which are represented by the unit normals $n_1$ and $n_2$ to the planes. We keep normals for left and right arms only, as our arms are more flexible and exert more complex motions which our legs cannot.

We represent vectors and normals with three angle cosines, and elbow and knee angles are also stores as cosine angles. Thus all features are in the same range of $[-1, 1]$, saving us the computation time of feature scaling. The four vectors $\{v_i\}$, four angles $\{\alpha_i\}$, and two normals $\{n_i\}$ constitute a 22-element feature vector.

## C. Smoothing Features

Due to real-world issues, such as the presence of obstacles, occlusion of body joints, and bad lighting conditions, skeleton data obtained from Kinect is often too noisy. Although Kinect already applies a filter to smooth raw skeleton data, after computing feature vectors from the skeleton frames, we see abrupt changes in feature values. In order to remove noise, we apply a 3rd order median filter [23], individually on each feature, to simultaneously remove noise and preserve edges on the curve. Figure 3 explains the process with an illustrative example.

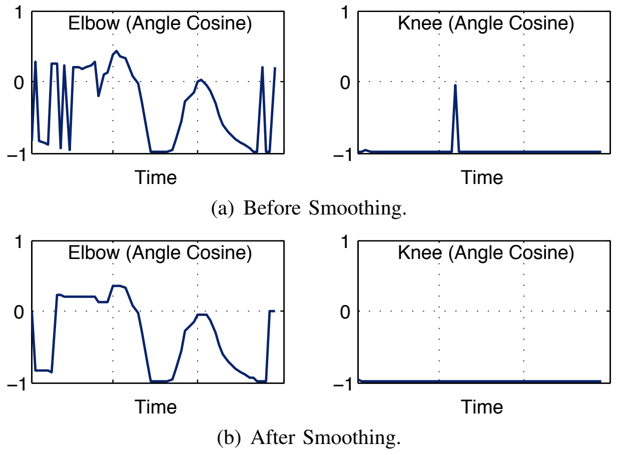

Fig. 3. Variations in angles at right elbow and knee during a 'hitting' action: (a) Elbow angles are noisy at the beginning and the end, and there is a spike in knee angles even though the person uses only his arm. (b) Filtering has made the data smoother and removed the spike.

## D. Dynamic Time Warping

The similarity between two feature vector sequences is computed using the dynamic time warping (DTW) algorithm, which is a well-known technique for computing the optimal alignment between two time-dependent sequences. DTW allows us to match an action with its faster or slower instances, and thus making our action recognizer invariant to the speed of an action.

Given two sequences of feature vectors $P := (p_1, p_2, \ldots, p_N)$ and $Q := (q_1, q_2, \ldots, q_M)$, where each of the $p_i$'s and $q_j$'s is a 22-element feature vector, the DTW distance is computed by the following recurrence relation:

$$D(n,m) = \begin{cases} \sum_{k=1}^{n} c(p_k, q_1), \text{for } n \in [1, N], m = 1 \\ \sum_{k=1}^{m} c(p_1, q_k), \text{for } m \in [1, M], n = 1 \\ \min\{D(n-1, m-1), D(n-1, m), D(n, m-1)\} \\ \quad + c(x_n, y_m), \text{for } 1 \le n \le N, 1 \le m \le M \end{cases}$$

where $c(x_i, y_j) = \|x_i - y_j\|$ is Euclidean norm, and the DTW distance is $D(N, M)$.

## E. Creating Classifiers

Although we could build a classifier that recognizes an unknown action by comparing its DTW distances from all labeled examples, such a recognizer would be extremely slow to be useful in practice. Instead, we use DTW as the kernel of a support vector machine (SVM). By default, an SVM uses a linear kernel (dot product of two vectors), however, it is not uncommon to see other types of non-linear kernels in use, including the DTW. Using DTW as a kernel allows us to have time-invariance as well as a fast and accurate classification at the cost of an increased training time. However, owing to the fact that the expert classifiers are not likely to be trained quite often, such a cost should be acceptable.

## V. Learning via User Feedback

In this section, first, we describe how labeled clusters are used to encode and classify an action, and then we describe how the clusters are obtained and the user's role in it.

## *A. Fixed-length Encoding of Skeleton Frames*

The $N$-length sequence of skeleton frames $s = (s_1, s_2, \dots, s_N)$, where $N$ is the window-size, is encoded to a $K$-length sequence $\zeta = (\zeta_1, \zeta_2, \dots, \zeta_K)$, before it is sent to a classifier. Let us assume that, we have a predefined set of vectors $C = \{\mu_j\}, 1 \le j \le T$, where each vector has the same dimensions as of a skeleton frame. The elements of $C$ are stick figures and the set as a whole serves as the code book for expressing the skeleton sequence using those stick figures. To obtain $\zeta$ from $s$, we do the following:

**Step 1:** Using the coordinate transformation matrix in Section IV-A, we obtain the sequence $b = (b_1, b_2, \dots, b_N)$ from sequence $s$.

**Step 2:** $\lfloor \frac{N}{K} \rfloor$ consecutive frames in $b$ are element-wise averaged to obtain $a = (a_1, a_2, \dots, a_K)$.

**Step 3:** Compute $\zeta_i = \underset{\mu_j \in C}{\text{argmin}} \|a_i - \mu_j\|, \text{for } 1 \le i \le K.$

## *B. Bayesian Classification*

Once we have the encoded sequence $\zeta$, we determine its most likely class $A_{NB}$ among all target classes $\{A_i\}$ using a Naive Bayes classifier:

$$A_{NB} = \underset{A_i}{\text{argmax}}\, p(A_i|\zeta_1, \zeta_2, \dots, \zeta_K) \tag{2}$$

$$= \underset{A_i}{\text{argmax}} \prod_{k=1}^{K} p(\zeta_k|A_i)p(A_i) \tag{3}$$

To evaluate the above equation, we assume a uniform prior, and use pre-computed $p(\zeta_k|A_i), 1 \le k \le K$, which are obtained by estimating the distribution of each $\zeta_k$ for each action, i.e., the fraction of times $\zeta_k$ assumes each of the $T$ values of $C$ for action $A_i$. Note that, the success of the classifier depends on the estimate of $p(\zeta_k|A_i)$. The more examples that are used to estimate this, the better is the classifier's accuracy. This is why this classifier needs more time than the expert classifier to evolve into one that has an acceptable recognition accuracy.

## *C. Cluster Creation and Update*

When actions are classified poorly, i.e. $p(A_{NB}|\zeta)$ is low, it indicates that the current codebook is not expressive enough to represent those actions properly. To handle this, we need to update the set of clusters, $C$. Let us assume, there are $T$ clusters and each of which is represented by a tuple, $(\mu_i, \sigma_i)$ where $\mu_i$ is the cluster-centroid and $\sigma_i$ is the mean point-to-centroid distance.

Recall that, all poorly classified examples are snapshotted and stored in a collection of size $M$ (Section III). Each element in this collection $\{a^i\}$ is denoted by the $K$-length representation, $a^i = (a^i_1, a^i_2, \dots, a^i_K)$ which is obtained by applying step 1 and 2 of the encoding algorithm described earlier. Let $\{a^i_j\}, 1 \le i \le M, 1 \le j \le K$, be the collection of all stick figures appearing in $\{a^i\}$. The following algorithm describes the cluster creation and update mechanism:

**Step 1:** Elements in $\{a^i_j\}$ are clustered using $k$-means algorithm with $k = T + 1$, producing $T + 1$ new clusters having their centroids at $\{\mu'_1, \mu'_2, \dots, \mu'_{T+1}\}$. For each new cluster with center $\mu'_i$, we repeat step 2.

**Step 2:** (a) If $\|\mu'_i - \mu_j\| \le 3\sigma_j$, we merge the $i^{th}$ new cluster with $j^{th}$ old cluster by updating $\mu_j$ to the weighted mean of $\mu'_i$ and $\mu_j$, and recomputing $\sigma_j$, (b) otherwise, we add $\mu'_i$ to $C$ and increment $T$.

Note that, creation and update of clusters neither changes the encoding length $K$ nor the representation $\zeta$ of previous actions. However, the distribution, $p(\zeta|A_i)$ changes as there are more examples now and $\zeta_k$ has zero or more new values to consider depending on the number of added clusters.

## *D. User Feedback*

To compute $p(\zeta|A_i)$, we first need to know what classes the new examples belong to. A naive approach is to let the user label each of the $M$ unknown examples, which is not feasible for large $M$. For this, we create a tree-structured hierarchical clustering of $M$ examples using agglomerative single-link clustering [16]. Then starting from the highest level, i.e. from the last link to the first, we ask the user if the two subgroups connected by the link are similar or not. If user says they are similar, we merge them and prompt the user to give it a name; otherwise, we recurse on each of its child links. Figure 4 illustrates this with a simplified case with $M = 7$ and $K = 2$.

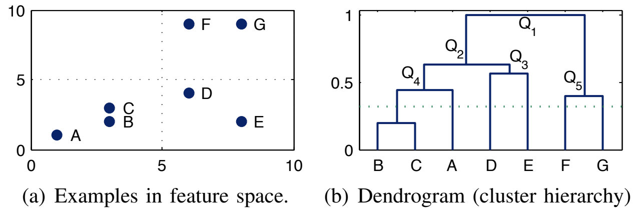

Fig. 4. A simplified example in a 2D-feature space showing how users are prompted to label clusters in a top-down manner.

Figure 4(a) shows the examples in their feature space, and Figure 4(b) shows the hierarchy (dendrogram). The highest level of the hierarchy divides the examples into two clusters, $\{A, B, C, D, E\}$ and $\{F, G\}$. The user is shown snapshots from each of them and is asked if they are similar. If he answers 'yes', we merge the clusters and give it a user given name. On the other hand, a 'no' means the clusters are different, and we need to recurse on each of the sets. The order of questioning is shown with $Q_i$ in the figure. We limit the recursion when a certain depth is reached (shown in dots). Examples below that depth (e.g. $\{B, C\}$) are too close and are assumed to be in the same cluster.

## VI. Empirical Evaluation of Expert Classifiers

This section describes a set of controlled experiments where we evaluate the accuracy, classification delay, and robustness of our expert classifiers. An overall system evaluation of Kintense is described in the next section.

We create an empirical dataset by collecting data from 19 users from 2 sites (UVA and DGIST, Korea) which is publicly available [4]. Each participant performs 4 actions (hitting, kicking, pushing, and throwing) plus some random actions such as jumping, waving hands, sit-ups, and clapping in front of a Kinect. We vary the relative distance (9 locations forming a $3 \times 3$ grid) and angle (7 angles: $0^\circ$, $\pm 20^\circ$, $\pm 40^\circ$, $\pm 60^\circ$) between the Kinect and the performer. Each action is performed $4 - 8$ times by a participant. In total, we have a dataset with about $13,000$ action instances. To the best of our knowledge, this is the largest publicly available dataset having skeleton data from Kinect with such a large number of participants, actions, and variations in distance and angles.

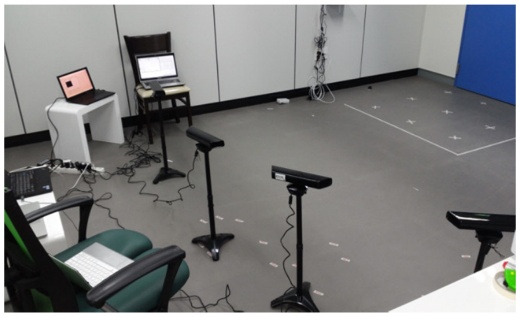
Fig. 5. A multi-Kinect setup to expedite the data collection.

We have two baselines. The first one is a DTW-based gesture matching algorithm for Kinect [3] which we modified to process 3D skeleton data from our dataset. The second one is a posture-based gesture recognizer, similar to [10], where we use sequences of stick figures as templates and use an SVM classifier. These two baselines are referred to as DTW and Pose, respectively. All the experiments are performed using Kinect sensors connected to a laptop having a 2.3 GHz Intel Core i5 processor and 4 GB RAM.

## A. Accuracy

We evaluate the accuracy of our expert classifiers and compare it with the two baselines for various amount of training. For each data point, we randomly select a subset of action instances from our dataset, train the algorithms using two-thirds of the data, and run tests on the rest. Each experiment is repeated $5 - 10$ times.

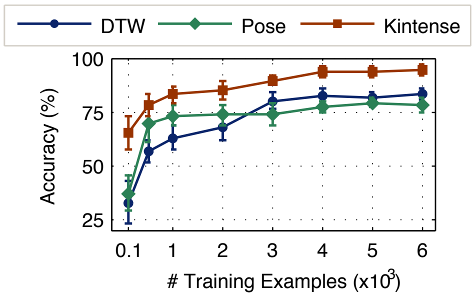

Fig. 6. Kintense is $11\% - 16\%$ more accurate than the baselines.

Figure 6 shows the accuracy of the 3 algorithms for training set sizes of $100 - 6000$ examples. The DTW, being a matching algorithm, performs the worst at the beginning, but later crosses Pose when it sees 2500 or more training examples. The accuracies of these two baselines do not rise much after 4000 examples, and converge to $83.3\%$ and $78.1\%$, respectively when we have the largest training set. Kintense, on the other hand, is always leading and its accuracy reaches $94.1\%$, which is $11\% - 16\%$ higher than the baselines.

## B. Classification Delay

We measure the time Kintense takes to process and classify each window of skeleton frames, and compare it to DTW and Pose algorithms. For Kintense, this includes pre-processing (e.g., coordinate transformation), feature extraction, smoothing, and SVM classification.

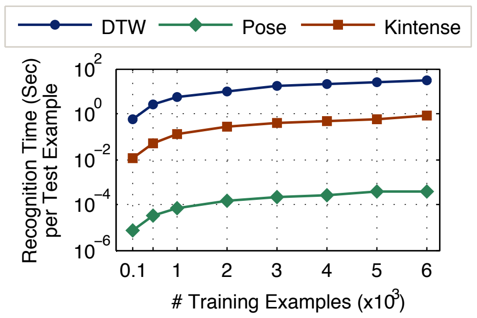

Fig. 7. Kintense shows an average delay of 336 ms which is higher than Pose, but is only $20\%$ of an average action-length.

Figure 7 shows the average classification time per test example for each of the 3 algorithms. We use a log-scale on the Y-axis since the classification time for DTW is up to $10^2 - 10^6$ times higher compared to the other two. The DTW is the slowest, as it compares each test example with every training example in the training set to find a match. Pose and Kintense, on the other hand, are offline-trained, and hence their classification time is much faster and both of them can be run in real-time. Among Kintense and Pose, Kintense is comparatively slower due to its expensive operations which results in a classification delay of about 336 ms. However, compared to the average duration of an action in our dataset, the delay is $< 20\%$. This means we can comfortably run Kintense in real-time even if multiple aggressive actions are performed back-to-back.

## C. Robustness

Unlike Kintense, none of the baselines explicitly handle robustness issues such as changes in distance, body orientation, speed of motion, and person. We describe a set of experiments to compare the robustness of the 3 algorithms.

*1) Distance:* To evaluate the robustness with respect to changes in distance, we measure the accuracy of each of the algorithms in two setups: (a) Fixed - the training and the test examples are from the same location, and (b) Varied - they are from different locations.

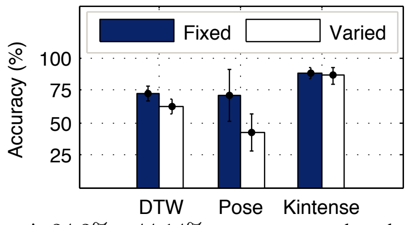

Fig. 8. Kintense is $24.2\% - 44.14\%$ more accurate than the baselines when they are trained and tested on examples from different locations.

Figure 8 shows the accuracy of each of the 3 algorithms for the two setups that we just described. We observe that, Kintense shows a great resistance with respect to changes in distance, sacrificing merely $2\%$ accuracy. On the other hand, Pose loses $27.2\%$ and DTW loses $10.6\%$ accuracy when the location is varied. In general, Kintense is $24.2\% - 44.14\%$ more accurate than the baselines when they are trained and tested on examples from different locations.

*2) Body Orientation:* To evaluate the robustness with respect to changes in body orientations, we measure the accuracy of each of the algorithms in 3 setups: (a) Zero - the person is facing the camera, (b) Fixed - the angle is fixed, and (c) Varied - trained and tested on two different angles.

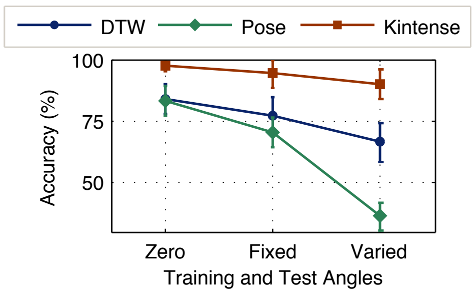

Fig. 9. Kintense is $23.8\% - 54\%$ more accurate than the baselines when the angle is varied.

Figure 9 shows the accuracy of each of the 3 algorithms for the three setups. The accuracy of DTW and Pose are similar when the user is facing the camera, but they are still $\sim 16\%$ less than Kintense's. The accuracy of all 3 algorithms drop at the fixed-angle case. This is because, when the angle is between $45° - 60°$, some body joints are occluded and it is difficult to locate them with Kinect. Kintense resists this to some extent, due to its feature smoothing capability and shows $16\% - 23\%$ more accuracy. When the training and test angles are different, we see a further reduction in accuracies. Especially, Pose becomes extremely inaccurate with only $36.1\%$ accuracy, while the DTW and Kintense show $66.2\%$ and $90\%$, respectively. Overall, Kintense is $23.8\% - 54\%$ more accurate than the baselines when the angle is varied.

*3) Speed:* To evaluate the robustness with respect to changes in speed of motion, we measure the accuracy of each of the algorithms in 3 setups: (a) Slow - both training and test examples are performed slowly, (b) Fast - performed fast, and (c) Mixed - trained and tested on different speeds. Using the timestamps of the first and last frame of an action, we computed the speed, sorted the actions based on speed, and then took the first one-third as slow and last one-third as fast actions.

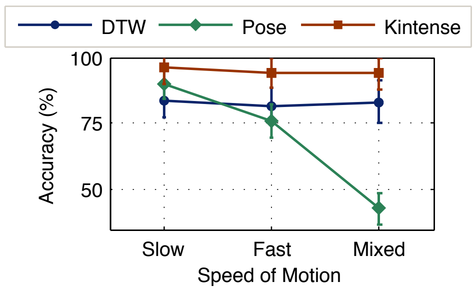

Fig. 10. Kintense is $10.9\% - 51\%$ more accurate than the baselines when the speed is varied.

Figure 10 shows the accuracy of each of the 3 algorithms for the three setups. We observe that, both DTW and Kintense are resilient to changes in speed, and their accuracies are similar in all 3 cases with Kintense showing $10.9\% - 12.43\%$ more accuracy than DTW. Pose, however, is slightly better than DTW for slow actions, but its accuracy sharply drops when the actions are performed fast and especially when we train and test it with actions of different speeds. Overall, Kintense is $10.9\% - 51\%$ more accurate than the baselines when the speed is varied.

*4) Person:* To evaluate the robustness with respect to changes in person, we measure the accuracy of each of the algorithms in two setups: (a) Fixed - the training and the test examples are from the same person, and (b) Varied - they are obtained from different persons.

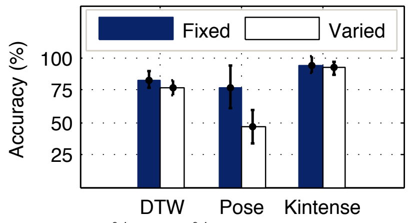

Fig. 11. Kintense is $16.14\% - 46.4\%$ more accurate than the baselines when the person is varied.

Figure 11 shows the accuracy of each of the 3 algorithms for the two setups. We observe that, both DTW and Kintense are more resilient to changes in person compared to Pose. DTW loses $6.5\%$ and Pose loses $31\%$ accuracy when they are trained and tested on different persons. Kintense, on the other hand, loses only $2.1\%$ accuracy and is overall $16.14\% - 46.4\%$ more accurate than the baselines when the person is varied.

## VII. Deployment and Full System Evaluation

To evaluate Kintense's performance in a real-world environment, we deploy the system in the living rooms of 2 two-person households, each having dimensions of approximately $12 \times 12$ square feet. The purpose of this experiment is to evaluate Kintense in a natural environment and with natural movements of the individuals, but we do identify the general script for a person to follow in order to perform the evaluation.

TABLE II. Performed Actions at Each Home

| Round # | Known Actions | Unknown Actions | Classes of Unknown Actions |
|---|---|---|---|
| 1 | $60 \times 4$ | 0 | none. |
| 2 | $60 \times 4$ | $60 \times 3$ | grab, poke, scratch. |
| 3 | $60 \times 4$ | $60 \times 5$ | grab, poke, scratch, resist, duck. |
| 4 | $60 \times 4$ | $60 \times 5$ | grab, poke, scratch, resist, duck. |

TABLE III. Description of Unknown Actions

| Action | Description |
|---|---|
| Grab | Grabbing action. |
| Poke | Poke own body (chest area) with a finger. |
| Scratch | Scratch head with a hand. |
| Resist | Waving a hand to show negativity. |
| Duck | Ducking action. |

The experiment is performed in multiple rounds. During each round, the two individuals of the household perform different aggressive actions, including the four actions that the expert classifiers are trained for. We ask them to perform each action with certain number of repetitions. However, they are not constrained to how to perform an action, what other actions to perform, the order of the actions, the speed, the duration, the body orientation, the distance from Kinect, etc. A human observer keeps records of the time and type of the actions they perform.

Kintense detects the actions it is trained for while identifying potential ones for human feedback. After each round, the operator is shown snapshots of potential aggressive actions from two different clusters and is asked to label them if they are from the same action. Once the labeling is done, the clusters are updated and the next round starts. Each round lasts for about $20 - 30$ minutes, and we stop after $4$ rounds.

Table II shows the number and type of actions performed by our participants at different rounds. The known actions refer to the classes our expert classifiers are trained for, and the unknown actions, which are described in Table III, are the ones that Kintense discovers and learns via user feedback.

## A. Accuracy and Evolution

After each round, we evaluate the accuracy of the expert classifier and the cluster-based classifier, and compute the overall accuracy of Kintense from the weighted average of the two. These are shown in Table IV.

TABLE IV. KINTENSE EVOLVES TO LEARN NEW ACTIONS AND TO IMPROVE ITS ACCURACY

| Round | Household 1 | | | Household 2 | | |
|---|---|---|---|---|---|---|
| # | Expert | Cluster | Average | Expert | Cluster | Average |
| 1 | 86.0% | - | 86.0% | 89.3% | - | 89.3% |
| 2 | 73.1% | - | 73.1% | 78.0% | - | 78.0% |
| 3 | 91.9% | 51.3% | 69.3% | 87.5% | 52.0% | 67.8% |
| 4 | 94.0% | 84.7% | 88.8% | 94.7% | 89.3% | 91.7% |

At round 1, the cluster-based classifier is not effective and hence the overall accuracy is the same as that of the expert's. At round 2, we introduce 3 new actions, and the accuracy of the expert drops by $11.3\% - 12.9\%$. But new clusters are created by Kintense to model them and they are recognized in the next round. At round 3, for the first time we see the cluster-based classifier's effect which has $51.3\% - 52\%$ accuracy as it recognizes only 3 out of 5 classes it has seen so far. The accuracy of the expert also rises as the examples that it misclassified previously are now handled by the cluster-based classifier. At round 4, the accuracy of cluster-based classifier, which now has the knowledge of all 5 classes, reaches $84.7\% - 89.3\%$, and the overall accuracy of Kintense becomes $\sim 90\%$. Note that, the apparent loss of accuracy in round 2 and 3 are transitional and is shown to demonstrate how Kintense catches up to its accuracy in the very next round.

In summary, Kintense accurately recognizes the aggressive actions of interest, and when new actions are introduced to it, the system automatically identifies them and lets a human operator decide if they are of any interest. And if he says they *are*, the system evolves to learn them, and correctly recognizes them in the very next round.

## B. Classification Delay

Table V shows the classification delays by both of the classifiers. As these classifiers run in parallel, we see an average delay of 345 ms, which is $\sim 20\%$ of a typical action's length. Hence, even when multiple actions are performed back-to-back, Kintense is able to detect and classify them in real-time.

TABLE V. CLASSIFICATION DELAY

| Classifier | Time (ms) |
|---|---|
| **Experts** | 297 |
| **Cluster-based** | 393 |

## C. False Alarms

We express the false alarms in terms of the false positive rate (FPR) which is the ratio of the false positives and the total number of positively classified examples by Kintense. These are shown in Table VI. We observe that, initially the system has a very low FPR of $7.3\% - 8.6\%$, but when $180$ unknown examples from $3$ unknown classes are introduced, it jumps to $24.7\% - 33.2\%$. About half of these are absorbed by the cluster-based classifier after round $3$, and after round $4$, the FPR becomes $6.4\% - 7.1\%$ as both the expert classifier and the cluster-based one become aware of all $9$ types of actions.

TABLE VI. FPR REDUCES AS THE SYSTEM EVOLVES AND LEARNS NEW ACTIONS.

| | Round 1 | Round 2 | Round 3 | Round 4 |
|---|---|---|---|---|
| **Household 1** | 7.3% | 24.7% | 13.6% | 6.4% |
| **Household 2** | 8.6% | 33.2% | 16.2% | 7.1% |

## D. User Feedback

Table VII shows the number of discovered actions and the amount of user interactions (i.e. queries) after each round. The first round having no unknown actions, the two values are zero. During the next two rounds, on average, Kintense discovers $203$ and $155$ action instances per household, corresponding to $180$ and $120$ newly introduced instances. The last round discovers $58$ more. A naive approach would require a human operator to individually tag each of these newly discovered instances. However, with the hierarchical clustering technique, Kintense reduces the amount of user interactions by $4.0 - 13.8$ times, and thus saves valuable work-hours of the human operator.

TABLE VII. KINTENSE REDUCES REQUIRED USER INTERACTIONS BY $4.0 - 13.8$ TIMES.

| Round | Household 1 | | | Household 2 | | |
|---|---|---|---|---|---|---|
| # | Discovered | Queries | Ratio | Discovered | Queries | Ratio |
| 1 | 0 | 0 | - | 0 | 0 | - |
| 2 | 207 | 15 | 13.8 | 198 | 22 | 9.0 |
| 3 | 166 | 18 | 9.2 | 144 | 20 | 7.2 |
| 4 | 26 | 6 | 4.3 | 32 | 8 | 4.0 |

## VIII. RELATED WORK

Several agitation detection techniques have been proposed in the past. Physiological signals, such as the galvanic skin response [20], [26], blood volume pressure [20], heart rate [26], skin temperature [26], and EEG waveforms [15] have been used to train models of agitating behavior. Studies on the acceleration of the wrists, ankles, and waist are performed to understand their association with agitation scale variables [6], [30]. The downside of these techniques is that, they require an active engagement of the person while collecting data – which is highly inconvenient. Compared to these techniques, Kintense is a completely non-invasive and passive monitoring system. Remote monitoring of sleeping patients for agitation has been performed using regular video cameras [12]. However, their downside is that they do not provide skeleton level information to accurately classify an action, and compared to video images, the stick figures from Kinect are far less privacy invasive.

Gesture recognition, specially hand gestures, from sequence of images are studied by many. [18] uses Hidden Markov Model (HMM) to spot and recognize hand gestures from a sequence of hand images. [8] uses tracking to recognize hand gestures. In our case, we get tracked skeleton information directly from the Kinect. Similar to Kintense, [34] uses location, angle and velocity as features, however, they solve a simpler problem of hand gesture recognition in 2D plane. [10] defines a gesture as a sequence of postures, recognizes poses using template matching and then uses an HMM to recognize gestures. This is similar to our pose-based baseline recognizer except that we use stick figures as templates.

Kinect has recently been used in several gesture recognition applications. Depth image from Kinect is used to recognize sign language [17], recognizing gaming actions [19], hand gestures [24], fall detection [28], gait analysis [29], predicting joint coordinates [27], and actions [32]. Our work is different from these as we use skeleton data whereas all these works have used only depth images from Kinect. However, skeleton data from Kinect has also been used in applications such as evaluating dance performance [5], mining actions [33], gait analysis [13], and simple gesture recognition [7]. However, unlike Kintense, none of these works consider issues such as the relative position, body orientation, and speed of motion of the person.

## IX. Conclusion

This paper describes Kintense, which is a robust, accurate, real-time, and evolving system for passively monitoring aggressive actions. Kintense employs a set of supervised learners to recognize well-known aggressive actions such as hitting, kicking, pushing and throwing with $11 - 16\%$ more accuracy and $10\% - 54\%$ more robustness to changes in distance, orientation, speed, and person than state-of-the-art techniques. Kintense employs semi-supervised learning to discover potential aggressive actions, and evolves itself to learn them with up to 13 times fewer user interactions than a typical user feedback based system. The evolving nature of Kintense makes it ideal for an agitation monitoring system where the actions to be detected are often not well-defined and the system needs to discover and evolve to learn them.

## References

[1] AceTek System. acetek.com.au/services/dementia-monitoring.

[2] Kinect SDK. http://kinectforwindows.org.

[3] Kinect SDK DTW Gesture Recognition. kinectdtw.codeplex.com.

[4] Kintense Dataset. http://tinyurl.com/cwy5a3x.

[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 *ACM Multimedia*, 2011.

[6] A. Bankole, M. Anderson, T. Smith-Jackson, A. Knight, K. Oh, J. Brantley, A. Barth, and J. Lach. Validation of noninvasive body sensor network technology in the detection of agitation in dementia. *American Journal of Alzheimer's Disease and Other Dementias*, 27(5):346–354.

[7] S. Celebi, A. S. Aydin, T. T. Temiz, and T. Arici. Gesture recognition using skeleton data with weighted dynamic time warping.

[8] F.-S. Chen, C.-M. Fu, and C.-L. Huang. Hand gesture recognition using a real-time tracking method and hidden markov models. *Image and Vision Computing*, 21(8):745–758, 2003.

[9] J. Cohen-Mansfield et al. Conceptualization of agitation: Results based on the cohen-mansfield agitation inventory and the agitation behavior mapping instrument. *International Psychogeriatrics*, 8(s 3):309–315.

[10] A. Elgammal, V. Shet, Y. Yacoob, and L. S. Davis. Gesture recognition using a probabilistic framework for pose matching. In *ICARCV 2002*.

[11] S. I. Finkel, J. S. Lyons, R. L. Anderson, et al. A brief agitation rating scale (bars) for nursing home elderly. *Journal of the American Geriatrics Society*, 41(1):50, 1993.

[12] F. Fook, P. V. Thang, T. M. Htwe, Q. Qiang, A. A. P. Wai, M. Jayachandran, J. Biswas, and P. Yap. Automated recognition of complex agitation behavior of dementia patients using video camera. In *e-Health Networking, Application and Services*, 2007.

[13] M. Gabel, R. Gilad-Bachrach, E. Renshaw, and A. Schuster. Full body gait analysis with kinect. In *EMBC*, pages 1964–1967, 2012.

[14] D. P. Hay et al. *Agitation in Patients with Dementia: a practical Guide to diagnosis and Management*. American Psychiatric Pub, 2008.

[15] G. Henderson, E. Ifeachor, N. Hudson, C. Goh, N. Outram, S. Wimalaratna, C. Del Percio, and F. Vecchio. Development and assessment of methods for detecting dementia using the human electroencephalogram. *Biomedical Engineering*, 53(8):1557–1568, 2006.

[16] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM computing surveys (CSUR)*, 31(3):264–323, 1999.

[17] A. Kurakin, Z. Zhang, and Z. Liu. A real time system for dynamic hand gesture recognition with a depth sensor. In *Signal Processing Conference (EUSIPCO)*, 2012.

[18] H.-K. Lee and J.-H. Kim. An hmm-based threshold model approach for gesture recognition. *IEEE PAMI*, 21(10):961–973, 1999.

[19] W. Li, Z. Zhang, and Z. Liu. Action recognition based on a bag of 3d points. In *IEEE CVPRW*, pages 9–14, 2010.

[20] W. Liao, W. Zhang, Z. Zhu, and Q. Ji. A real-time human stress monitoring system using dynamic bayesian network. In *IEEE CVPRW*, pages 70–70, 2005.

[21] T. Monahan and T. Wall. Somatic surveillance: Corporeal control through information networks. *Surveillance & Society*, 4(3), 2002.

[22] V. Patel and R. Hope. A rating scale for aggressive behaviour in the elderly–the rage. *Psychological medicine*, 22(01):211–221, 1992.

[23] W. K. Pratt. *Digital Image Processing*. John Wiley & Sons, 1978.

[24] D. Ramirez-Giraldo, S. Molina-Giraldo, A. M. Alvarez-Meza, G. Daza-Santacoloma, and G. Castellanos-Dominguez. Kernel based hand gesture recognition using kinect sensor. In *STSIVA*, 2012.

[25] A. Rosenblatt. The art of managing dementia in the elderly. *Cleveland Clinic journal of medicine*, 72(Suppl 3):S3, 2005.

[26] G. E. Sakr, I. H. Elhajj, and H.-S. Huijer. Support vector machines to define and detect agitation transition. *Affective Computing, IEEE Transactions on*, 1(2):98–108, 2010.

[27] J. Shotton, T. Sharp, A. Kipman, A. Fitzgibbon, M. Finocchio, A. Blake, M. Cook, and R. Moore. Real-time human pose recognition in parts from single depth images. *Communications of the ACM*, 56(1):116–124.

[28] E. E. Stone and M. Skubic. Evaluation of an inexpensive depth camera for passive in-home fall risk assessment. In *PervasiveHealth*, 2011.

[29] E. E. Stone and M. Skubic. Passive in-home measurement of stride-to-stride gait variability comparing vision and kinect sensing. In *Engineering in Medicine and Biology Society*, pages 6491–4, 2011.

[30] T. Tamura, T. Fujimoto, and T. Togawa. Quantitative assessment of behavior in dementia patients by continuous physical activity monitoring. In *Engineering in Medicine and Biology Society*, 1997.

[31] T. Voisin, S. Andrieu, C. Cantet, and B. Vellas. Predictive factors of hospitalizations in alzheimers disease: a two-year prospective study in 686 patients of the real. fr study. *The journal of nutrition, health & aging*, 14(4):288–291, 2010.

[32] J. Wang, Z. Liu, J. Chorowski, Z. Chen, and Y. Wu. Robust 3d action recognition with random occupancy patterns. In *ECCV*. 2012.

[33] J. Wang, Z. Liu, Y. Wu, and J. Yuan. Mining actionlet ensemble for action recognition with depth cameras. In *CVPR*, 2012.

[34] H.-S. Yoon, J. Soh, Y. J. Bae, and H. Seung Yang. Hand gesture recognition using combined features of location, angle and velocity. *Pattern Recognition*, 34(7):1491–1501, 2001.