Knowledge Engineering for Unsupervised Canine Posture Detection from IMU Data

Michael Winters
North Carolina State University
Raleigh, NC, USA
mawinter@ncsu.edu

Rita Brugarolas
North Carolina State University
Raleigh, NC, USA
rbrugar@ncsu.edu

John Majikes
North Carolina State University
Raleigh, NC, USA
jjmajike@ncsu.edu

Sean Mealin
North Carolina State University
Raleigh, NC, USA
spmmealin@ncsu.edu

Sherrie Yuschak
North Carolina State University
Raleigh, NC, USA
sherrie_yuschak@ncsu.edu

Barbara Sherman
North Carolina State University
Raleigh, NC, USA
barbara_sherman@ncsu.edu

Alper Bozkurt
North Carolina State University
Raleigh, NC, USA
aybozkur@ncsu.edu

David Roberts
North Carolina State University
Raleigh, NC, USA
dlrober4@ncsu.edu

ABSTRACT
Training animals is a process that requires a significant investment of time and energy on the part of the trainer. One of the most basic training tasks is to train dogs to perform postures on cue. While it might be easy for a human trainer to see when an animal has performed the desired posture, it is much more difficult for a computer to determine this. Most work in this area uses accelerometer and/or gyroscopic data to produce data from an animal’s current state, but this has limitations. Take for example a normal standing posture. From an accelerometer’s perspective, it closely resembles the “laying down” posture, but the posture can look very different if the animal is standing still, versus walking, versus running, and might look completely different from a “standing on incline” posture. A human trainer can instantly tell the difference between these postures and behaviors, but the process is much more difficult for a computer.

This paper demonstrates several algorithms for recognizing canine postures, as well as a system for building a computational model of a canine’s potential postures, based solely on skeletal measurements. Existing techniques use labeled data, which can be difficult to acquire. We contribute a new technique for unsupervised posture detection, and compare the supervised technique to our new, unsupervised technique. Results indicate that the supervised technique performs with a mean 82.06% accuracy, while our unsupervised approach achieves a mean 74.25% accuracy, indicating that in some cases, our new unsupervised technique is capable of achieving comparable performance.

Author Keywords
Animal-computer interaction; Classification algorithms; canine training; accelerometers; animal-machine interfaces

ACM Classification Keywords
Algorithms; Design; Hardware

INTRODUCTION
Dogs are trained to perform a wide range of behaviors in their roles as military working dogs, search and rescue dogs, protection dogs, guide dogs for the blind, and so on. The training process for these animals can be time-intensive, requiring the work and supervision of knowledgeable and experienced animal trainers.

One of the most fundamental training tasks is teaching an animal to perform a desired posture on cue. Many approaches to training use positive reinforcement—that is, rewarding the animal when it has performed the desired action. However, animals may incorrectly associate a different behavior with the reward—this is called superstitious behavior. For example, if you want to train your dog to sit, you might say the word (or perform a visual cue), then coax the dog into the position, then give the dog a treat, repeating this process over
and over. However, the dog may also look away from you (in order to see what your hands are doing), and may then learn that sitting and looking away is the desired behavior. Or worse, the dog may learn that simply looking away is the desired behavior (which in this example is definitely not the case).

Computer-assisted animal training can be useful for a number of the applications previously mentioned. Human trainers can be inconsistent both with their recognition of the correct animal behavior and with the prompt delivery of the reward. Both of these are crucial for developing an efficient training protocol [2, 15]. By incorporating computers into the process, we aim to make training more accurate and more efficient, potentially allowing for animals to be trained more rapidly, or for more complex behaviors. However, automating the training process is complex, and requires a number of different components. Going back to the previous example, if algorithms can be developed to detect not only the desired postures and behaviors, but any extraneous, superstitious behaviors as well, then the reinforcement of those unwanted behaviors can be reduced or eliminated.

In this paper, two techniques for building models for canine posture detection are presented. The first technique is a supervised learning technique, and requires labelled training data, while the second technique only requires skeletal measurements from the animal. In addition, three approaches for classifying canine postures based on those models are also presented.

Since collecting and labelling training data is a laborious process (for these experiments, video recordings were made and then analyzed manually) it is not easy or practical outside of a lab setting. A knowledge engineering approach allows us to completely forego most or all of the data collection/algorithm training process. Furthermore, since our goal is to eventually automate the animal training process, the system needs to be able to recognize postures and behaviors that the animal does not currently know how to perform on cue, and therefore collecting training data for those postures and behaviors is especially difficult.

Our eventual goal is to be able to develop a system where an average pet owner can take a few simple skeletal measurements of an animal, put our harness on the animal, and have an accurate posture recognition system with no more effort than that. We want to take as much of the burden of training the animal out of the hands of the human and put it into a hardware and software system.

Data was collected from five different dogs to evaluate the performance of the three different posture recognition techniques on a range of postures. Results indicate that our classification algorithms can perform quite well in many cases, but do have room for improvement. Furthermore, our new approach for using skeletal measurements to build a model of a canine’s postures, while not as accurate as the tried and true supervised learning method, provides a reasonable degree of accuracy, and establishes a baseline for further development.

RELATED WORK

The idea of using accelerometers or inertial measurement units to measure activity (whether for humans or canines) is not new [9, 12, 13, 14], but there are many challenges that have yet to be addressed for dealing with activity and posture detection in canines.

Prior work in this area has explored the optimal number and location of sensors [5], which demonstrated that a chest-mounted and back-mounted accelerometer produced high information gain for the postures they analyzed.

Work has also been done to evaluate different algorithms for posture recognition [4, 6], which have evaluated cascading machine learning algorithms (using random forest, k-nearest neighbors, and logistical model trees), as well as hidden Markov models. Accelerometer data has also been used to identify seven static postures and dynamic behaviors [8], however the system did not include wireless capability and data was stored on local memory for off-line analysis. Another aspect of this work is that they didn’t use the heuristic features of accelerometers but instead computed 126 features, unnecessarily increasing the complexity.

PROCESS

There were several components to our posture recognition system. First, we designed hardware to be worn by dogs that enables positional measurements to be recorded. Second, we created a way to collect and label the data with the correct postures and build a computational model based on this labelled training data. Lastly, new algorithms were designed to recognize when the animal was in a given posture. Here each of those components is described in further detail.

Hardware

Our harness was built using a LightBlue Bean [7], developed by Punch Through Design. This small device contains an onboard 3-axis accelerometer, and we connected a second accelerometer (LSM303) to the bean by a short cable. Previous work has shown that the optimal position for two accelerometers is on the animal’s lower back (near the hips), and under the dog’s chin/neck area [5]. The harness design underwent several iterations, eventually becoming a small, lightweight “harness” that consisted of a collar and a belly band, each holding a single accelerometer (see Figure 1). Prior work used a much larger, heavier, bulkier harness, as well as a heavier and more powerful onboard computing platform [3]. While the earlier version of the harness was powerful and modular, it could only be worn by dogs of a sufficient size, weight, and strength. Furthermore, many dogs require time or training in order to acclimate to a large, bulky harness. Our goal with this "miniaturized" version of the hardware was to create a smaller, lighter system that could be worn by dogs of any size, and would be less intrusive or uncomfortable for the animal. This system could easily be adapted to quadrupeds of any size, from small cats to large horses.

One downside to using the LightBlue Bean is the very limited processing power available. The bean has an Arduino ATmega 328p processor running at 8MHz. In practice, our data collection program could only run at approximately 2Hz (as compared to 10Hz in prior work [4, 5, 6]). It is believed
that the bottleneck for this program is the Bluetooth to serial communication protocol used by the bean. Furthermore, the algorithms presented in this paper were designed specifically with this hardware in mind. In previous work, processing power was more freely available, so it was possible to use more complex algorithms, and also to collect more detailed data. In this experiment, only six axes of accelerometer data (X,Y,Z from each of the two different accelerometers), were collected (prior work also used gyroscopic data). One of the goals with this experiment was to build accurate and efficient models and classification algorithms that could work with fewer sensor data and on smaller hardware.

Dog Interaction & Data Collection Process

For this study, five dogs were recruited from contacts within the local animal training community. Our criteria for inclusion in the study was that the animal needed to be capable of performing the “sit” posture on cue, and be able to hold postures for at least three to five seconds. We were looking for as much variety as possible in terms of the sizes of the dogs and the number of additional postures each animal could perform.

In order to collect data, the harness was put on the dog with the owner’s assistance. At the same time, a video camera was recording the dog’s actions. The owner then cued the dog to stand and sit (as well as any other postures the dog could perform), holding each posture for several seconds. Some dogs were able to perform more postures than others. We also observed “natural” postures, such as laying down. Once the data was collected, the video was manually synchronized with the data logs, and we used the video recording to manually label the data that had been collected. This was a time-consuming process, which is a large part of why we are working to eliminate the need to collect this training data entirely.

In addition to collecting and labeling posture data, skeletal measurements were collected from each dog. These data, as well as a visual representation of the measured areas, can be seen in Table 1 and Figure 2, respectively. For each animal tested, the following skeletal measurements were collected: spine (from neck to base of tail (A)), the height of the shoulder (from tip of humerus to ground (B)), front upper leg (humerus (C)), front lower leg (radius/ulna (D)), front paw (carpus/metacarpus/phalanx (E)), the height of the hip (from tip of femur to ground (F)), back upper leg (femur (G)), back mid leg (tibia/fibula (H)), back lower leg (tarsus/metatarsus (I)), back paw (phalanx (J)), and weight. All measurements were taken while the dog was standing in a normal, relaxed posture. All protocols and procedures involving dogs were approved by the Institutional Animal Care and Use Committee.

Evaluation

Several methods were used for evaluating both the data and the classification algorithms. As a baseline for comparison, the random forest classification algorithm [11] was used. We also used two different techniques for building the posture models. The first technique was a supervised learning approach, while the second technique was a knowledge-engineered approach. For each model creation technique, every permutation of the three decision rules across all six sensor inputs was tested. All told, this amounted to 4096 different decision rule schemes (three rules plus exclusion for each of the six axes yields $4^6 = 4096$ possible schemes).

Random Forest Algorithm

The random forest algorithm was used as a comparative baseline, since it has proven to be highly accurate for this type of data [4, 6]. For this experiment, we used the Weka 3 Data Mining software, and ran a 10-fold cross validation of our data using the random forest algorithm [1].

Model-based Classification

The classification approach we developed requires a “posture model” as input. This posture model is a range of numeric values for each sensor showing what values are applicable for a given posture. A simplified visualization of this can be seen in Figure 3.

Supervised Model

With the hardware in place, we could stream the positional data from the accelerometers back to a computer to be
Table 1. Dog skeletal measurements (in inches). Locations for the measurements are depicted in Figure 2. All dogs were male.

<table>
<thead>
<tr>
<th>Dog</th>
<th>Spine (A)</th>
<th>Shoulder height (B)</th>
<th>Front upper leg (C)</th>
<th>Front lower leg (D)</th>
<th>Front paw (E)</th>
<th>Hip height (F)</th>
<th>Back upper leg (G)</th>
<th>Back mid leg (H)</th>
<th>Back lower leg (I)</th>
<th>Back paw (J)</th>
<th>Weight (K)</th>
<th>Breed</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>26</td>
<td>18</td>
<td>8</td>
<td>8</td>
<td>3</td>
<td>18</td>
<td>9</td>
<td>6</td>
<td>5</td>
<td>3</td>
<td>58</td>
<td>Australian Shepherd</td>
</tr>
<tr>
<td>D2</td>
<td>18</td>
<td>13</td>
<td>7</td>
<td>6</td>
<td>2</td>
<td>14</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>2</td>
<td>26</td>
<td>Beagle</td>
</tr>
<tr>
<td>D3</td>
<td>24</td>
<td>21</td>
<td>10</td>
<td>10</td>
<td>2.5</td>
<td>21</td>
<td>9</td>
<td>9</td>
<td>5</td>
<td>2.5</td>
<td>52</td>
<td>Border Collie</td>
</tr>
<tr>
<td>D4</td>
<td>15</td>
<td>8.5</td>
<td>3.5</td>
<td>4</td>
<td>1.5</td>
<td>9</td>
<td>5</td>
<td>3.5</td>
<td>3</td>
<td>1.5</td>
<td>16.7</td>
<td>Poodle &amp; Dachshund mix</td>
</tr>
<tr>
<td>D5</td>
<td>32</td>
<td>20</td>
<td>9</td>
<td>9</td>
<td>3</td>
<td>22</td>
<td>8</td>
<td>9</td>
<td>7</td>
<td>3</td>
<td>70</td>
<td>German Shepherd</td>
</tr>
</tbody>
</table>

Figure 3. Visualization of three axes of a posture model. Note the relatively small “window” sizes, and the significant overlap between windows. These are the X, Y, Z windows for dog D5, for the back-mounted accelerometer.

Table 2. The set of postures we built skeletal models for. Note that not all dogs performed the same postures.

<table>
<thead>
<tr>
<th>Label</th>
<th>Description</th>
<th>Performed by</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>Standing</td>
<td>D1, D2, D3, D4, D5</td>
</tr>
<tr>
<td>S</td>
<td>Sitting</td>
<td>D1, D2, D3, D4, D5</td>
</tr>
<tr>
<td>E</td>
<td>Eating</td>
<td>D2, D3</td>
</tr>
<tr>
<td>B</td>
<td>Bowing</td>
<td>D1</td>
</tr>
<tr>
<td>VS</td>
<td>Vertical stand/beg</td>
<td>D3, D4</td>
</tr>
<tr>
<td>L</td>
<td>Lay down</td>
<td>D1, D2, D3, D4</td>
</tr>
<tr>
<td>LL</td>
<td>Lay on left side</td>
<td>D1, D3, D5</td>
</tr>
<tr>
<td>LR</td>
<td>Lay on right side</td>
<td>D4</td>
</tr>
<tr>
<td>LHL</td>
<td>Lay halfway on left side</td>
<td>D1, D3, D5</td>
</tr>
<tr>
<td>LHR</td>
<td>Lay halfway on right side</td>
<td>None</td>
</tr>
<tr>
<td>LB</td>
<td>Lay on back</td>
<td>None</td>
</tr>
</tbody>
</table>

recorded and analysed. This gave us acceleration values on three axes (X, Y, Z) from two locations on the animal’s body, for a total of six points of data at each time step (for the purposes of our algorithms, each input can be thought of as a distinct sensor). We then added a seventh point of data, the label for the animal’s current posture. This labelling was done manually, using synchronized video as a reference. Next, a model based on the postural data was created. To build the model, the program grouped all seven inputs by label and then calculated the mean and standard deviation of six accelerometer axes. This was done independently for each dog.

The model for a given posture, then, is a range of values centered around the mean ± a certain number of standard deviations. A model includes ranges of values associated with each of the six accelerometer readings. A larger number of standard deviations gives us a larger window. A simplified visualization of this model can be seen in Figure 3.

To determine the optimal size of these windows, we compared the performance of 36 different models created from 0.0 to 3.5 standard deviations in increments of 0.1. While the optimal number of standard deviations to use for the model generation varied from one dog to another, the best performance tended to be between 1.0 and 1.5 standard deviations. A value of 1.1 standard deviations was chosen for future model generation, since it seemed to be a good compromise across dogs and algorithms.

Knowledge Engineered Model
The bigger goal of our work was not just to do posture recognition for postures dogs already know how to perform (that is—postures that we had collected data and built a model for), but for untrained postures as well (that is—postures which we had not collected data and built a model for). This will allow us to eventually automate the training process for postures that an animal does not yet currently know how to execute on cue. To do this, we needed a computational understanding of how the animal moves.

We identified a series of eleven possible postures that the animal could be in (see Table 2). For the LHL and LHR postures, the dog’s upper body would be lying flat on the floor, while the lower body is twisted sideways. We then created a computational model of what the six sensor values should look like for each posture, based on the animal’s skeletal measurements. We created a new model-generating program that would take the animal’s skeletal measurements as input data (and nothing else—no training data was required with this approach), and create a new model based on those measurements.

For many of the postures, the skeletal measurements actually had no bearing on the location and size of the window. For example, all of the “lay down” variants, including the “lay on
sideline” postures, are not affected by the animal’s height, spine length, etc, since those skeletal measurements do not affect the angle that the accelerometers sit at in those particular positions. However, one of the key postures we looked at, the “sit” posture, was dramatically affected by both the animal’s height and spine length. We found that when transitioning from a stand to a sit that the hip height of the animal dropped to approximately 20% of it’s original value. Using geometry, we can then calculate the angle of the animal’s spine while sitting down. As an example, Algorithm 1 shows how that process was implemented for the sit posture. Similar approaches were used for the other postures presented in Table 2, but the postures most reliant on skeletal measures are sit, eat, and bow.

Algorithm 1 Sit posture calculation
1: # hh = hipHeight
2: # sl = spineLength
3: # an accelerometer value of ±260 is equivalent to 90 degrees
4: yCenter = (90-atan(hh - (hh*.2)/sl)/90) * -260
5: zCenter = (atan(hh - (hh*.2)/sl)/90) * 260
6: yLow = yCenter - 1.1σ
7: yHigh = yCenter + 1.1σ
8: zLow = zCenter - 1.1σ
9: zHigh = zCenter + 1.1σ
10: # analysis of collected data showed that the window size (at 1.1σ) for sit was approximately 150 units for most dogs

Windows for postures such as vertical stand, bow, and eat were computed in a similar fashion, while windows for the laying postures were generated by manual analysis of the previously collected data. The various laying postures look almost identical across dogs, regardless of size. With the new skeletal models created, they were evaluated using the same decision schemes used with the supervised models.

Decision Rules
We tested several different decision rules for using the models to classify postures, and calculated and ranked the accuracy of each. By doing this, we were able to not only see which techniques were more accurate at recognizing postures, but also which data fields were more or less crucial. This information can allow us to save the more complex algorithms only for the data fields that were most important, and to use the faster but less accurate algorithms on the other data fields, in order to save compute cycles on the low-power embedded hardware being worn by the animal.

Window Match
Our most basic approach, the Window Match (WM or W), would only recognize a posture if the input data fell completely within the “window” specified by the model. If no posture was matched, a posture of “Unknown” was selected. We experimented with requiring that all six sensors agree on a posture classification, but quickly discovered that this was highly inaccurate. Since the harness could sometimes shift on the animal during normal movement, a slight shift from the position that the model was built could render the model ineffective with this approach. We subsequently modified it to have each sensor select a posture individually, and counted the number of “votes” for each posture. If a single posture had a majority of the votes, then that posture was selected.

Clustering
The Clustering (C) decision rule improves upon the Window Match by selecting the closest posture for a given input. Where the Window Match would only select a posture if the input value was completely inside the window as defined by the model, the Clustering rule uses the distance between the current input and the centers of each posture window described by the model, and then selects the closest one. This behaves similarly to the K-means clustering algorithm [10]. This was able to compensate for shifts in the harness caused by the animal’s movement. Each sensor was given an equal “vote,” as in the Window Match, and if a single posture received a majority of votes, then that was the posture that was returned.

Fuzzy Clustering
The Fuzzy Clustering (FC or F) decision rule improves upon the Clustering decision rule by calculating the distance to the closest outer edge of each neighboring window, as well as a value indicating how close the input is to that window border. With all algorithms previously described, each sensor would get discrete “votes” for various postures. For example, the X1 sensor would return all postures that matched for that sensor, and so on for every other sensor. In the Fuzzy Clustering algorithm, those “votes” are no longer discrete. In addition to each sensor determining what postures matched for the given input data, a floating point score or weight would be calculated for each posture. A score of 1.0 would be a full vote, identical to how the previous algorithms operated, while a value between 0.0 and 1.0 would be a partial vote. The weight was scaled linearly between the two closest windows, so if an input value fell inside a window, it received a weight of 1.0, and the weight decreased linearly as it got further outside the boundary. A weight of 0.0 would be centered between the two closest window boundaries.

Once all sensors had performed their individual calculations, the tallies for each posture were summed, as well as the scores. As with the previous algorithms, the highest scoring posture would be the one that is selected.

The fuzzyness of this algorithm allowed for situations where some sensors might have a weak preference for a given posture, while other sensors might have a very strong preference for a different posture. In the Clustering rule, situations like this could result in a tie, but Fuzzy Clustering was able to overcome this deficiency. This was also intended to better handle overlapping postures, such as “standing” versus “laying down,” which have very similar acceptance windows. It was often the most accurate algorithm, but it was also the slowest, particularly given the fact that it involved floating point math, which is much slower than integer math on the Arduino-based processor that the LightBlue Bean has.

Results
In this section we present the results of our analysis of the learned and skeletal models in comparison to the random forest classifier. We also present data illustrating how performance would be affected using only a single accelerometer.

**Learned Models**
Table 3 shows the accuracy of using the window match decision rule (WM) for all axes, the clustering decision rule (C) for all axes, and the fuzzy clustering decision rule (FC) for all axes, with a posture model built using 1.1 standard deviations. In addition, this table shows the accuracy for the “best average” decision rules, as well as the best rules for each individual dog. In the case of the “best average” rules, all of the axes were ranked for each dog individually, and then the average rank for each was calculated across all dogs. By doing this, we can see which rule scheme (or schemes, since there were several ties) performed well across all dogs. The best average performance is in the second-to-last column of Table 3. The list of best average algorithms is in Table 4. The final column in Table 3 shows the best algorithms for that dog. The list of best algorithms for each dog is in Table 5.

For dogs D1, D2, and D5, the random forest algorithm performed equal to or only slightly better than the other techniques (for learned models). In the case of D5, there were 346 out of 4096 rule schemes that outperformed the random forest algorithm. However, the random forest algorithm performed notably better than the other schemes for dogs D3 and D4. A careful inspection of the data showed that these dogs had a much greater amount of extraneous movement, causing the input data to be noisier. This also made the manual labelling process more difficult and less accurate. In previous experiments, the random forest algorithm tended to produce accuracies in the mid to high 90% range [6], so seeing results this low is an indicator that the training data was not especially high quality.

**Skeletal Models**
To see the accuracy data for the skeletal models, see Table 6. In the case of dog D3, the skeletal model actually performed better than the learned model. This dog also exhibited the largest number of distinct postures. The skeletal models for the remaining dogs did not perform as well as their learned/trained counterparts, with the learned models performing approximately 10% to 15% higher than the skeletal models.

**Accuracy when using a single accelerometer**
Another question we wanted to explore is how badly accuracy was degraded by using only a single accelerometer. We ran all data through all permutations of only having three axes worth of input data. When using only the data from the collar mounted accelerometer, accuracy dropped tremendously, with the highest accuracy being in the low 20% range, which is far too low to be acceptable in actual use. Accuracy using only the back-mounted accelerometer is shown in Tables 7 & 8. Interestingly, the WM and FC rules were both equally successful when used for all three axes with the skeletal models (for most of the dogs), but the FC rule was typically the best for the learned models. The skeletal models tended to have slightly larger windows, which allowed the WM to accurately classify more data than with the smaller windows of the learned models.

**FUTURE WORK**
Since this work was exploratory in nature, each owner was asked to cue the dog to demonstrate as many different postures as possible. However, the number of postures that each dog performed varied, and that may have been a factor in making our classification algorithms less accurate or less consistent. We would like to conduct another series of experiments with a more rigorous set of postures that all dogs must perform, and for longer durations than was done during these experiments.

Additionally, we would like to develop a method to quickly and accurately determine what the best performing algorithm combinations will be in real-time, or perhaps if a deeper understanding is achieved of how each algorithm is affected by various factors of the models, the optimal algorithm combination can be determined when the model is generated. If we continue to use low-power embedded hardware such as the Bean, the optimal algorithm combination will need to be determined when the model is built.

Since the unsupervised method (using only skeletal measurements) exhibited a significant accuracy drop compared to the supervised method, we would like to explore the creation of a semi-supervised model, which will begin with the skeletal model as a foundation, and then allow a human trainer to tune and adjust the model with a reduced training period (compared to the current supervised method).

Lastly, we would like to further refine our current skeletal model generation system. With more posture data from a larger number of animals, we hope to get a more thorough understanding of what factors affect the window sizes, and in what ways.

**CONCLUSION**
Several new classification schemes for canine posture detection, as well as two techniques for building posture models have been presented. The first technique involved collecting a set of labelled training data, while the second technique only required skeletal measurements from the animal. The models built using the first technique were more accurate, but also much more time-consuming to create (including a laborious data collection and labelling process), while the models built using skeletal measurements required very little effort from the animal handler.

While the supervised technique achieved a mean accuracy of 82.06%, the unsupervised technique achieved a mean accuracy of 74.25%. There are many ways that the algorithms presented in the paper could be improved upon, however, they present a foundation upon which future work can be built; the ultimate goal being the complete automation of the canine training process.

**ACKNOWLEDGMENTS**
This material is based upon work supported by the National Science Foundation under Grant No. 1329738. Any opinion,
Table 3. Accuracy of decision rule schemes using learned models in comparison to the random forest classifier. The decision rule schemes used the same decision rule for all six axes as a point of comparison to the best performing of the 4096 possibilities, both for the individual dog and averaged across all dogs. The specific best performing rules can be found in Tables 5 & 4.

<table>
<thead>
<tr>
<th>Dog</th>
<th>Num Postures</th>
<th>Acc (RF)</th>
<th>Acc (WM)</th>
<th>Acc (C)</th>
<th>Acc (FC)</th>
<th>Acc (Best Avg)</th>
<th>Acc (Best)</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>6</td>
<td>95.79%</td>
<td>89.47%</td>
<td>89.47%</td>
<td>92.1%</td>
<td>91.58%</td>
<td>92.63%</td>
</tr>
<tr>
<td>D2</td>
<td>4</td>
<td>82.48%</td>
<td>79.56%</td>
<td>68.61%</td>
<td>81.02%</td>
<td>73.72%</td>
<td>82.48%</td>
</tr>
<tr>
<td>D3</td>
<td>7</td>
<td>85.71%</td>
<td>50.29%</td>
<td>52%</td>
<td>52.57%</td>
<td>56%</td>
<td>60%</td>
</tr>
<tr>
<td>D4</td>
<td>5</td>
<td>85.18%</td>
<td>55.56%</td>
<td>58.33%</td>
<td>50.93%</td>
<td>76.85%</td>
<td>77.78%</td>
</tr>
<tr>
<td>D5</td>
<td>4</td>
<td>93.59%</td>
<td>91.03%</td>
<td>92.31%</td>
<td>93.59%</td>
<td>94.87%</td>
<td>97.43%</td>
</tr>
<tr>
<td>Mean</td>
<td>5.2</td>
<td>88.55%</td>
<td>73.18%</td>
<td>72.14%</td>
<td>74.04%</td>
<td>78.60%</td>
<td>82.06%</td>
</tr>
</tbody>
</table>

Table 4. Best algorithms (averaged across all dogs)

<table>
<thead>
<tr>
<th>Model type</th>
<th>Best Algorithms</th>
<th>Rank (D1)</th>
<th>Rank (D2)</th>
<th>Rank (D3)</th>
<th>Rank (D4)</th>
<th>Rank (D5)</th>
<th>Rank (averaged)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Learned</td>
<td>WCCFFW</td>
<td>3</td>
<td>13</td>
<td>8</td>
<td>2</td>
<td>3</td>
<td>5.8</td>
</tr>
<tr>
<td>Skeletal</td>
<td>WFFFCN</td>
<td>2</td>
<td>12</td>
<td>3</td>
<td>14</td>
<td>17</td>
<td>9.6</td>
</tr>
</tbody>
</table>

findings, and conclusions or recommendations expressed in this material are those of the authors(s) and do not necessarily reflect the views of the National Science Foundation.

REFERENCES


Table 5. Best algorithms for each individual dog. Here, “W” is the Window Match algorithm, “C” the Clustering algorithm, and “F” the Fuzzy Clustering algorithm, while “N” is the Null algorithm, which ignored the input for that axis. The order is X1, Y1, Z1, X2, Y2, Z2, where the back-mounted accelerometer is the first three, and the collar-mounted accelerometer is the last three.

<table>
<thead>
<tr>
<th>Model type</th>
<th>D1</th>
<th>D2</th>
<th>D3</th>
<th>D4</th>
<th>D5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Learned</td>
<td>WCCNFC, CCCNFC, CFFNFC, FCCNFC, FFCCNF, FCCCCN, FFFFFC</td>
<td>CFFFNW, FFFWFF, FFFFWF, FFFFFF</td>
<td>FCFCFNN</td>
<td>WCWCWN, WCFWNF, WFFCWN</td>
<td></td>
</tr>
<tr>
<td>Skeletal</td>
<td>WWWCCN, WWFCCN, WFWCCN, FWFCCN, FFWCCN, FFFFFC</td>
<td>WWWWWN, WWWFWN, WFWFWF, WWFWFN, WWFWWF</td>
<td>WNNWWCF, WNCWFC, FWFCWN, FFFCWN</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 6. Accuracy of decision rule schemes using skeletal models in comparison to the random forest classifier. The decision rule schemes used the same decision rule for all six axes as a point of comparison to the best performing of the 4096 possibilities, both for the individual dog and averaged across all dogs. The specific best performing rules can be found in Tables 5 & 4.

<table>
<thead>
<tr>
<th>Dog</th>
<th>Num Postures</th>
<th>Acc (RF)</th>
<th>Acc (WM)</th>
<th>Acc (C)</th>
<th>Acc (FC)</th>
<th>Acc (Best SK Avg)</th>
<th>Acc (Best)</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>6</td>
<td>95.79%</td>
<td>67.36%</td>
<td>50.52%</td>
<td>67.36%</td>
<td>78.95%</td>
<td>80.52%</td>
</tr>
<tr>
<td>D2</td>
<td>4</td>
<td>82.48%</td>
<td>67.88%</td>
<td>11.68%</td>
<td>67.15%</td>
<td>61.31%</td>
<td>69.34%</td>
</tr>
<tr>
<td>D3</td>
<td>7</td>
<td>85.71%</td>
<td>53.71%</td>
<td>23.42%</td>
<td>54.86%</td>
<td>66.86%</td>
<td>68%</td>
</tr>
<tr>
<td>D4</td>
<td>5</td>
<td>85.18%</td>
<td>37.96%</td>
<td>39.81%</td>
<td>34.26%</td>
<td>48.15%</td>
<td>61.11%</td>
</tr>
<tr>
<td>D5</td>
<td>4</td>
<td>93.59%</td>
<td>76.92%</td>
<td>16.67%</td>
<td>75.64%</td>
<td>71.95%</td>
<td>92.31%</td>
</tr>
<tr>
<td>Mean</td>
<td>5.2</td>
<td>88.55%</td>
<td>60.77%</td>
<td>28.42%</td>
<td>59.85%</td>
<td>65.44%</td>
<td>74.25%</td>
</tr>
</tbody>
</table>

Table 7. Accuracy of decision rule schemes using learned models and only a single (back-mounted) accelerometer. The decision rule schemes used the same decision rule for all three axes as a point of comparison to the best performing of the 64 possibilities for each dog.

<table>
<thead>
<tr>
<th>Dog</th>
<th>Num Postures</th>
<th>Accuracy (RF)</th>
<th>Accuracy (WM)</th>
<th>Accuracy (C)</th>
<th>Accuracy (FC)</th>
<th>Accuracy (Best)</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>6</td>
<td>94.74%</td>
<td>79.47%</td>
<td>50.52%</td>
<td>67.36%</td>
<td>78.95%</td>
</tr>
<tr>
<td>D2</td>
<td>4</td>
<td>78.10%</td>
<td>45.99%</td>
<td>11.71%</td>
<td>67.15%</td>
<td>61.31%</td>
</tr>
<tr>
<td>D3</td>
<td>7</td>
<td>75.93%</td>
<td>34.86%</td>
<td>23.42%</td>
<td>54.86%</td>
<td>66.86%</td>
</tr>
<tr>
<td>D4</td>
<td>5</td>
<td>70.86%</td>
<td>37.96%</td>
<td>39.81%</td>
<td>34.26%</td>
<td>48.15%</td>
</tr>
<tr>
<td>D5</td>
<td>4</td>
<td>94.87%</td>
<td>76.92%</td>
<td>16.67%</td>
<td>75.64%</td>
<td>71.95%</td>
</tr>
<tr>
<td>Mean</td>
<td>5.2</td>
<td>82.9%</td>
<td>51.01%</td>
<td>28.42%</td>
<td>59.85%</td>
<td>65.44%</td>
</tr>
</tbody>
</table>

Table 8. Accuracy of decision rule schemes using skeletal models and only a single (back-mounted) accelerometer. The decision rule schemes used the same decision rule for all three axes as a point of comparison to the best performing of the 64 possibilities for each dog.

<table>
<thead>
<tr>
<th>Dog</th>
<th>Num Postures</th>
<th>Accuracy (RF)</th>
<th>Accuracy (WM)</th>
<th>Accuracy (C)</th>
<th>Accuracy (FC)</th>
<th>Accuracy (Best)</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>6</td>
<td>94.74%</td>
<td>58.95%</td>
<td>5.26%</td>
<td>58.95%</td>
<td>58.95%</td>
</tr>
<tr>
<td>D2</td>
<td>4</td>
<td>78.10%</td>
<td>52.44%</td>
<td>1.46%</td>
<td>52.44%</td>
<td>52.44%</td>
</tr>
<tr>
<td>D3</td>
<td>7</td>
<td>70.86%</td>
<td>48.57%</td>
<td>0.57%</td>
<td>48.57%</td>
<td>48.57%</td>
</tr>
<tr>
<td>D4</td>
<td>5</td>
<td>75.93%</td>
<td>38.89%</td>
<td>3.70%</td>
<td>38.89%</td>
<td>38.89%</td>
</tr>
<tr>
<td>D5</td>
<td>4</td>
<td>94.87%</td>
<td>74.36%</td>
<td>0.0%</td>
<td>74.36%</td>
<td>74.36%</td>
</tr>
<tr>
<td>Mean</td>
<td>5.2</td>
<td>82.9%</td>
<td>56.56%</td>
<td>2.20%</td>
<td>56.75%</td>
<td>56.75%</td>
</tr>
</tbody>
</table>