Automatic behavior sensing for a bomb-detecting dog

Hoa G. Nguyen*, Adam Nans, Kurt Talke, Paul Candela, H.R. Everett
Space and Naval Warfare Systems Center Pacific
San Diego, CA 92152

ABSTRACT

Bomb-detecting dogs are trained to detect explosives through their sense of smell and often perform a specific behavior to indicate a possible bomb detection. This behavior is noticed by the dog handler, who confirms the probable explosives, determines the location, and forwards the information to an explosive ordnance disposal (EOD) team.

To improve the speed and accuracy of this process and better integrate it with the EOD team's robotic explosive disposal operation, SPAWAR Systems Center Pacific has designed and prototyped an electronic dog collar that automatically tracks the dog's location and attitude, detects the indicative behavior, and records the data. To account for the differences between dogs, a 5-minute training routine can be executed before the mission to establish initial values for the *k-mean clustering* algorithm that classifies a specific dog's behavior.

The recorded data include GPS location of the suspected bomb, the path the dog took to approach this location, and a video clip covering the detection event. The dog handler reviews and confirms the data before it is packaged up and forwarded on to the EOD team. The EOD team uses the video clip to better identify the type of bomb and for awareness of the surrounding environment before they arrive at the scene. Before the robotic neutralization operation commences at the site, the location and path data (which are supplied in a format understandable by the next-generation EOD robots—the Advanced EOD Robotic System) can be loaded into the robotic controller to automatically guide the robot to the bomb site.

This paper describes the project with emphasis on the dog-collar hardware, behavior-classification software, and feasibility testing.

Keywords: bomb detection, canine, robot, k-mean clustering, electronic dog collar, behavior sensing

1. INTRODUCTION

Bomb-detecting dogs often perform off-leash (see Figure 1) to identify potential explosives through their sense of smell, then sit or lie down, depending on how they have been trained, to indicate a possible bomb detection. The dog handler, within sight of the dog, notes this detection behavior. Based on experience and observation of the surrounding environment, he then decides whether this is a probable bomb site, and determines its location through various means. This information is then relayed back to base or to the explosive ordnance disposal (EOD) team.

This reporting process is slow and inaccurate due to the ad hoc nature of the location determination and communication. We sought to improve the speed and accuracy of this process by equipping the dog with sensors and automated behavior-detection algorithms. In addition, we are not limiting the dog to a specific type, as the search dog may belong to canine teams from various branches of the law-enforcement or armed forces. This wide applicability means that the dogs may have been trained differently and the concepts of operation may vary, as for example:

- The dog may lie or sit when potential explosives have been detected.
- The dog may or may not look at the detected explosive device.
- The dog may be directed from one (possibly false) detection to search for other explosives without returning to the handler.

* hoa.nguyen@navy.mil

Figure 1. Typical patrol with off-leash military working dog.[1]

The information we needed to collect from the dog included: GPS location of the dog as it exhibited the bomb-detection behavior, the route the dog took to arrive at the event location, and a video clip prior to and continuing for a period of time after the detection. The dog's track may indicate a safe approach route; and the video clip would aid the EOD team in visualizing the operating environment and the type of bomb in advance, and further confirming the location of the bomb from various landmarks.

If the data conforms to the format used by the EOD team's robot controller, it will further simplify the subsequent neutralization team's effort. The dog's track and bomb location can be loaded into the robot controller, displayed on the same map, and used to drive the robot to the location of the bomb. We chose to use the data format of the Multiple-robot Operator Control Unit (MOCU),[2] which is the operator control software of the next-generation EOD robots, the Advanced EOD Robotic System (AEODRS).[3] MOCU also allows the robot to automatically approach the bomb location using the supplied track as waypoints.

Although this paper presents the bomb-detection application, the capability described can be adapted to other operational concepts, and is not necessarily limited to explosives detection and disposal.

2. OBJECTIVES AND CONSTRAINTS

In order to formulate the objectives and determine the constraints for the project, we met with several dog handlers and an EOD team, all with recent field experience in theater. From these discussions, the objectives were narrowed down and constraints formulated as follows.

Objectives:

1. Facilitate the transmission of accurate location of potential bomb detected by dog handler to EOD team.
2. Improve the location accuracy of the potential explosive device.
3. Provide to the EOD team the track that the dog took to arrive at the potential bomb.
4. Provide to the EOD team useful video clips that can help identify the type or location of the explosive device.

Constraints:

1. Emit no RF radiation near potential bomb site. (Although it is unlikely that an RF signal from the dog near an explosive device would set it off, it was determined that some dog handlers would not use the system if it emits RF.)
2. Provide a simple system that does not complicate the job of the operator.

3. Keep added hardware (dog's or dog handler's) to a minimum.
4. Provide a versatile system that can handle different types of working dog, i.e., the dog may lie or sit down when it detects an explosive device, depending on how it was trained.

3. APPROACH

In this section, we will describe the hardware and software developed to achieve the objectives while staying within the constraints stated above. We will focus on the dog-collar hardware and detection software, and touch briefly on the software running on the dog handler's handheld computer that will be used to filter the possible detections, and the EOD team's software that will make use of the data generated by the dog-collar.

3.1 Dog-collar hardware

The dog-collar hardware was chosen to be small, light-weight, and low cost. The collar had to be small enough to not impact the dog's ability to move his head comfortably, and light-weight enough that the dog was not bothered by the additional hardware. These objectives were accomplished by using a small-form-factor processor, camera, IMU, and GPS receiver, connected as shown in Figure 2.

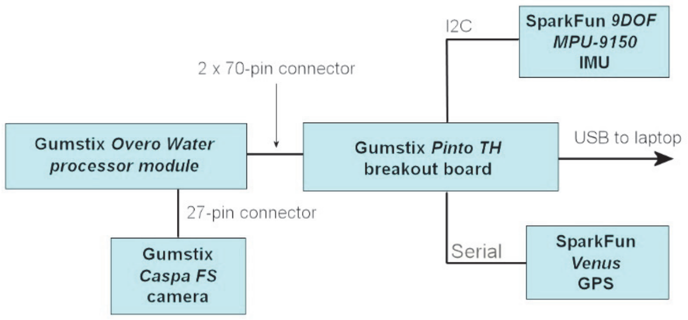
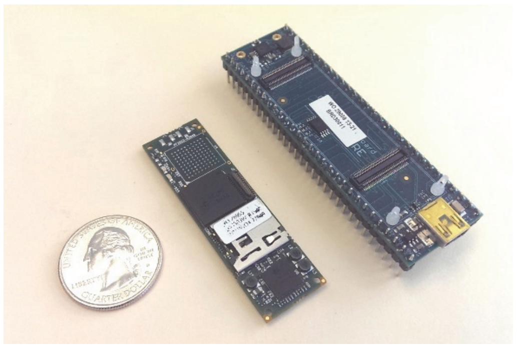


Figure 2. Dog-collar system interconnect diagram.

Several processor boards were considered, ranging from microchip micro-processors, Raspberry Pi modules, to Gumstix computers. The Gumstix *Overo Water* computer-on-module (COM) was ultimately chosen because of its small form factor and its interfaces. The Gumstix *Overo Water,* with the *Pinto-TH* expansion board, provides 2 serial ports, an I2C line, and a USB connection exposed for easy integration with other sensors. The *Pinto-TH* breakout board also offers voltage regulation at 1.8v and 3.3v that was used by the other sensors in the collar. Other versions of the Gumstix exist that offer Bluetooth and WiFi, but the *Overo Water* was chosen because of the absence of these options. Part of the design of this collar was to reduce the amount of excess RF energy emitted by the collar to reduce the possibility of triggering an explosive device accidentally. Figure 3 shows the Gumstix processor module, the *Pinto-TH* break-out board, and a U.S. quarter-dollar coin (for scale).



Gumstix *Overo Water*

- 720 MHz processor
- 200 MB RAM
- C64x DSP chip
- Mini SD card slot

Figure 3. The Gumstix *Overo Water* processor module and the *Pinto-TH* break-out board. A US quarter is shown for size comparison.

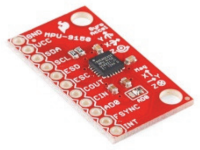
The camera chosen for the dog collar was the Gumstix *Caspa FS* (see Figure 4). This camera was selected for its simple integration with the Gumstix and the small form factor, but it did not perform as hoped. There were problems with the software driver for this device on many of the newer Gumstix kernels, and the integration of the camera and software proved to be challenging. The camera was not able to do any auto-focusing, and the capture rates were less than 10 frames a second. In the future a different camera would be recommended—the function of the collar could have been accomplished by a USB camera to possibly better effect.

Gumstix *Caspa FS* camera

- 640 x 480 resolution
- Full spectrum with infrared
- 39mm x 25.7 mm

Figure 4. The Gumstix *Caspa FS* camera and characteristics.

The SparkFun *9DOF MPU-9150* motion-tracking module was chosen because of its small form factor, low cost, and 3-axis acceleration measurements (see Figure 5). The module contains the InvenSense *MPU-9150* micro-electro-mechanical-system (MEMS) IMU. Acceleration sensitivity was set for a range of +/- 2 G, which is large enough to prevent the motions of the dog from saturating the sensor, yet small enough to detect small motions of the dog at rest. This IMU was well within functional ranges for the collar, however, one with a different interface may be recommended for production purposes. During testing, any noise introduced on the I2C line would cause the processor and this chip to get out of sync, and the processor was unable to recover from the error. If there was either another device on the line to correct for the I2C collisions, or the driver on the Gumstix was written to correct for these errors, this device would be fine. As it exists, an IMU with a serial interface may be a better match for the system.



SparkFun *9DOF MPU-9150*

- Tri-axis accelerometer
- Tri-axis gyroscope
- Tri-axis magnetometer
- Motion fusion

Figure 5. The SparkFun IMU unit and characteristics.

The SparkFun *Venus* GPS board (Figure 6) was chosen because of its small form factor, reported accuracy, and quick startup/time-to-first-fix (TTFF). It uses the SkyTraq *Venus63LPx* GPS chip. In testing, the GPS solution stayed within the reported accuracy ranges, but the TTFF seemed to be significantly longer than the advertised 29 seconds. This may be a result of the location of the testing, but the simple interface and accuracy made this device adequate for our prototype.

SparkFun *Venus* GPS

29 s TTFF cold start
2.5m accuracy
WAAS / EGNOS support

Figure 6. The SparkFun GPS board and characteristics.

The final design of the collar uses three separate boxes for the components to more evenly distribute the weight on the dog's neck (see Figure 7). The battery box houses rechargeable batteries, battery-protection circuits, and the power switch. The weight of the battery box is somewhat offset by the GPS and IMU in the box on the opposite side of the collar. The processor and camera are in the center housing, which also has the connection points for downloading data from the collar when the dog handler reaches a stopping point. The six pins on top of the center box allow for a charging-and-communication cable to be attached to interact with the collar and charge its battery. The weighting and shapes were designed to keep the camera and main processor centered under the dogs head, rather than allowing it to rotate around the neck of the dog as it moves.

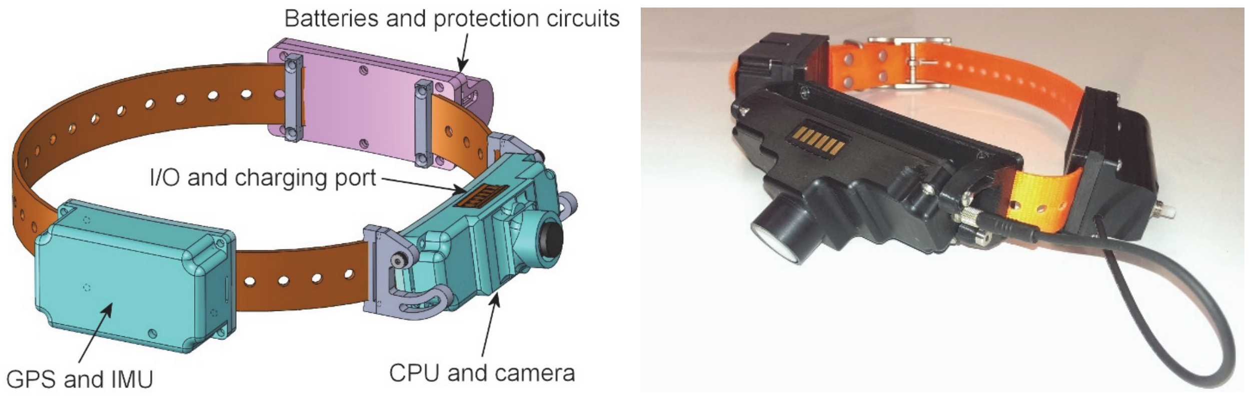


Figure 7. CAD rendering of the dog collar (left) and final prototype made with milled plastic enclosures.

3.2 Attitude-detection software

The software running on the Gumstix processor on the collar is responsible for tracking the position of the dog as it does it search, determining when the dog is exhibiting the behavior indicative of a detection (trigger condition), and recoding the video for the approach and trigger. The GPS positions are stored every second as the dog is out, and cached for 90 seconds. On a trigger, the GPS points are stored into an XML file that is associated with this detection. Similarly the video is cached in a circular buffer, old frames are pushed out as new frames come in. On a positive trigger the frames that are stored in the buffer are recorded, and additional frames are captured. This gives us a video clip for the time leading up to the detection and a little after the detection.

The principal task for the dog-collar software is to determine when the dog is in a detection position. The software relies on a few assumptions for this task. Since the dog has been trained to sit or lie down upon a detection, the first the assumption is that while the dog is searching, the range of accelerations experienced by the collar will be noticeably more than when the dog is in a detection position. The second assumption is that the dog will behave in a similar fashion for each detection. The final assumption is that the dog will remain in the detection position for at least 2 seconds.

Finding the detection condition is accomplished by first filtering the raw IMU acceleration data to remove large spikes in measurements caused by noise, hard foot falls, or as the dog comes down from a jump. Filtering is performed by choosing the median data point within a sliding window 1-second wide (10 data points). The filtered data is then subjected to a *k-means clustering* algorithm to classify the data and detect trigger conditions.

K-Means Clustering[4]

K-means clustering attempts to group sets of data into a number of clusters where the error from any measurement to the center of the cluster is minimized for each cluster. This means that the distance from the centroid of the cluster to any of the data within is less than the distance from the data to the centroid of another cluster. When clustering it can be helpful to know how many clusters are in the data set, and what they mean. For the assumptions used by the collar we needed three clusters of data. These clusters are: (1) Detection behavior: minimum change in accelerations, (2) Searching behavior: range of change in accelerations, and (3) Extreme behavior: large changes in accelerations. Originally we only used two clusters: Detection and Search. After some time, however, the searching cluster was pulled too high by the existence of extreme motions (even with filtering). This prompted the creation of the third cluster to help keep normal motion at appropriate levels.

The process for *k-means clustering* is demonstrated below and in Figure 8:[4]

1. Chose initial cluster (represented by a centroid) position (discussed further below).
2. Move points into the closest clusters

$$S_i^{(t)} = \left\{x_p : \left\|x_p - m_i^{(t)}\right\|^2 \le \left\|x_p - m_j^{(t)}\right\|^2 \forall j, 1 \le j \le k\right\} \tag{1}$$

where S denotes a cluster, k is the maximum number of clusters, x is each data point, and m is a centroid.

3. Update centroids

$$m_i^{(t+1)} = \frac{1}{\left|S_i^{(t)}\right|} \sum_{x_j \in S_i^{(t)}} x_j \tag{2}$$

4. Repeat steps 2 and 3 until the clusters (centroids) no longer move.

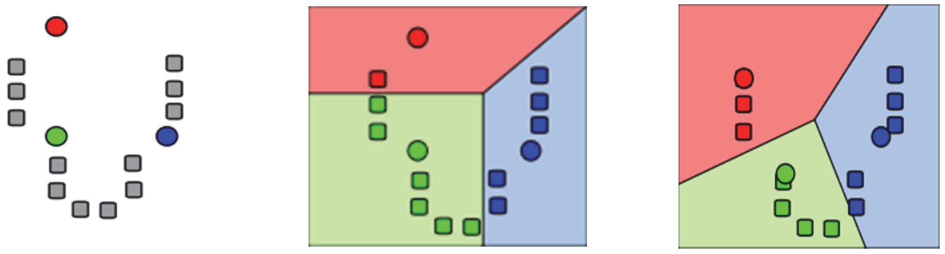


Figure 8. The *k-means clustering* process (see steps 1 to 3 above).[4]

<u>Choosing initial clusters</u>

There are many ways to choose the initial values for the clusters. One method is to use multiple random initial locations and see where they converge. Another way is to keep adding more clusters in the hopes that the number of clusters in the data was more than the number chosen. For our pose-monitoring algorithm, the initial cluster positions are chosen to be in meaningful locations based on the definitions of the clusters, as follows:

1. Detection (minimum changes in acceleration)

   This cluster was initialized to be at the minimum changes in acceleration found through testing.

2. Search (normal changes in acceleration)

   This cluster was set to be initialized at the mean change in acceleration. This ensures that the searching cluster stays around the middle of the data.

3. Jarring/extreme motion (high changes in acceleration)

   This cluster was initialized to the maximum change in accelerations.

Before the algorithm is used for actual data classification, it needs to be trained so that the cluster centroids better match the particular dog. This is accomplished by having the dog wear the collar for about 10 minutes while it is led through the various activities (running, walking, sitting, lying), which could be incorporated into exercise or play time. Then as the collar is used in actual classification/detection activities, the software continues to make small adjustments to the clusters as more data is collected. The initial training is automatically performed if no saved data exists in the dog-collar's non-volatile memory, which can be reset when the collar is put on a different dog.

Figure 9 show an example of data collected by the collar. The jarring/extreme motion is shown by the spikes in the graph, and a few detection sections are shown as the valleys. There are some points that fall within the detection clusters but did not create a trigger event. This is because a detection event requires 2 consecutive seconds of data in the detection-value range (i.e., the dog must remain in the sitting or lying position for at least 2 seconds).

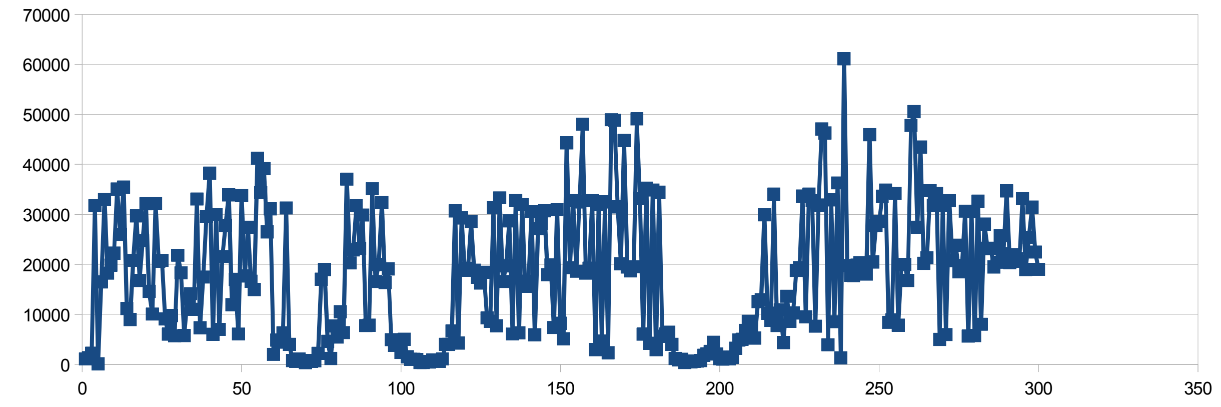


Figure 9. Example of data collected from the dog collar. The *x* axis represents time, and the *y* axis acceleration.

3.3 Dog-handler's software

As soon as the collar electronics determine that a possible detection event has occurred, the current GPS location and the route and video data from the last 30 seconds (pulled from the continuously running circular buffer), plus an additional 30 seconds of video taken after the trigger, are frozen and stored in a file marked with the time stamp of the detection. Occasionally, the dog handler would recall the dog to him (perhaps during a break), snaps a connector onto the dog collar, and download the files to his own wearable computer, laptop, or handheld computer. Running a program that is a subset of the EOD MOCU software, he can review the stored data files by scrolling through the list, viewing the routes displayed on a map and the associated video clips. He can then decide to either keep a file as a valid detection, discard it, or mark it as a possible detection. For files considered valid or possible detections, he can use a slider to mark the beginning and ending of a segment in the video that he thinks is important (see Figure 10). That information will be packaged with the rest of the data and saved for forwarding to the EOD team**.**

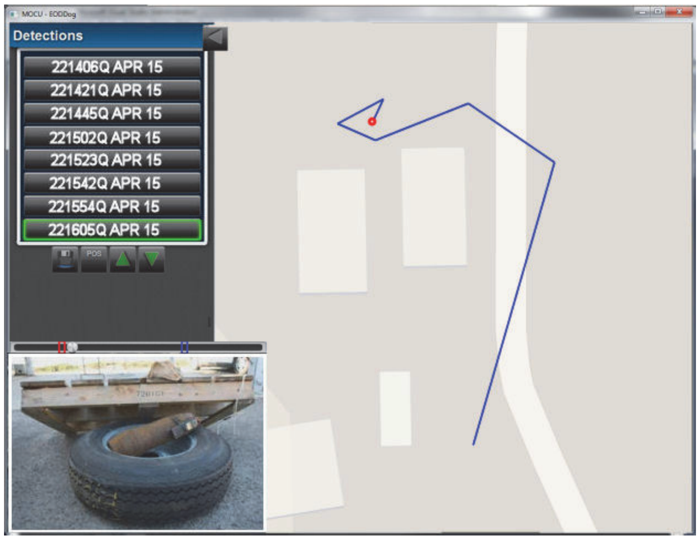


Figure 10. Representative dog-handler's computer screen, showing path and video of selected time-stamped file. The video is being edited, with the beginning and ending points of a segment of interest marked.

3.4 EOD team's software

Once the EOD team receives the data files from the dog handler, the robot operator can import the files into the MOCU program running on the robot controller (normally a ruggedized laptop). They will be displayed on the screen, as shown in Figure 11. The operator can then elect to view the detections or possible detections, examine the videos, and command the AEODRS robot to automatically follow the path stored in the data file to reach the selected bomb site.

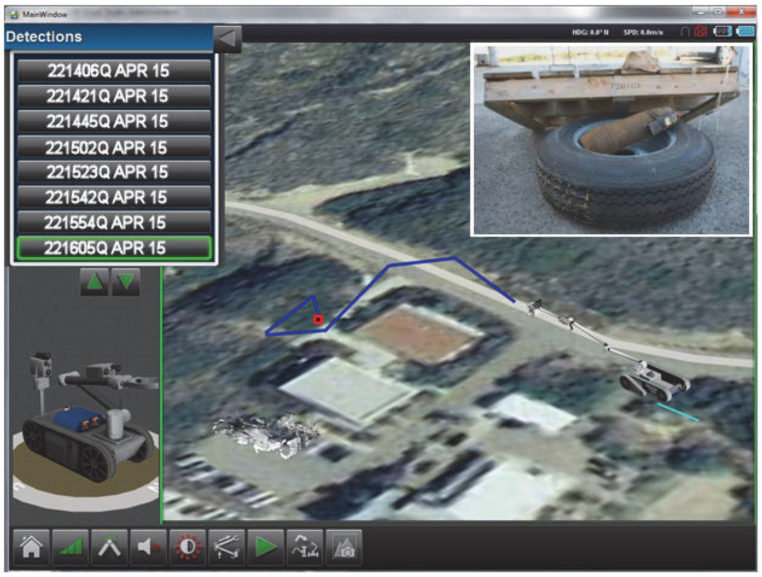


Figure 11. Representative MOCU screen showing the time-stamped video of and waypoint path to the suspected bomb.

4. DEMONSTRATIONS

Three demonstrations were conducted with three different dogs at two locations with a range of terrain. One location had large open spaces with a mix of trees and fields. The other was in a canyon with large hills on both sides and sections with elevation changes. The three dogs were not bomb-detection dogs and had different levels of training, but all had the basic ability to sit on command. Figure 12 shows two of the dogs wearing the electronic collar. We noted the time whenever the dogs were commanded to sit, then later compared those values to the time recorded by the dog collar for the detections.

We found that the collar performed fairly accurately, with only 1 missed detection and 2 false positives for 23 commanded "sit" events. The missed detection was probably due to the fact that these dogs were not trained to sit absolutely still to indicate a detection, as is done with bomb-detection dogs,[5] but continued to look around even when they have been commanded to sit. The two false positives came from a large, slow, and lumbering dog (Figure 12, right) that did not generate large differences in acceleration between the walking and sitting states. These false positives could be easily dismissed by the handler during the review process described in Section 3.3.

We believe that the results would have been much better if we had used dogs that were specifically trained to perform bomb-detection. Possible improvement could also come from additional sensor inputs in the *k-means clustering*, such as the orientation of the collar with respect to the gravity vector (available from the IMU). However, it is unclear how much orientation information will help, as we have found that the orientation of the dog's neck does not differ much between the sitting/lying and walking poses.

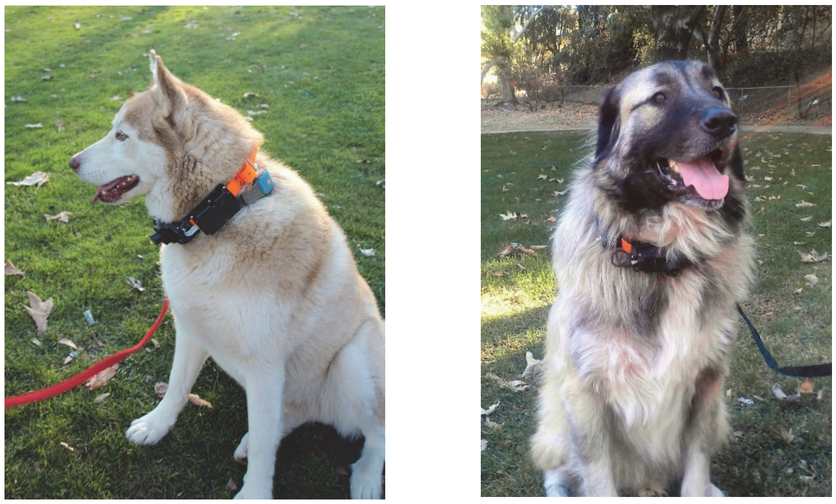


Figure 12. The electronic collar on two of the dogs used in the demonstrations. The dogs were also wearing their normal collar and leash.

5. SUMMARY

We have developed a prototype electronic dog collar that automatically detects specific behaviors of a trained dog through classification of acceleration data. *K-means clustering* is used to separate the data into classes corresponding to states of rest, normal movement, and extreme movements. Detection of certain behavior is then registered based on the length of time the dog spends in a state. Although we applied this technique to automate the recognition of the trained behavior of bomb-detection dogs as they detect possible explosives, the capability described can readily be adapted for other animal-behavior recognition tasks.

ACKNOWLEDGEMENTS

This project was supported by the Office of Naval Research, Expeditionary Maneuver Warfare and Combating Terrorism Department (Code 30).

REFERENCES

[1] Photo source: US Marine Corps.
[2] D. Powell, D. Barbour, and G. Gilbreath, "Evolution of a common controller," Proc. SPIE 8387: Unmanned Systems Technology XIV, Baltimore, MD (2012).
[3] M.A. Hinton, M.J. Zeher, M.V. Kozlowski, and M.S. Johannes, "Advanced Explosive Ordnance Disposal Robotic System (AEODRS): A Common Architecture Revolution," Johns Hopkins APL Technical Digest, 30(3), (2011).
[4] "K-means clustering," Wikipedia, < http://en.wikipedia.org/wiki/K-means_clustering> (24 March 2015).
[5] "Detection Dog Training 'Passive Alert'," YouTube, <https://www.youtube.com/watch?v=XRJkacnRDEM> (8 April 2015).