

Sparse Detector Sensor: Profiling Experiments for Broad-Scale Classification

D. J. Russomanno*, M. Yeasin, E. Jacobs, M. Smith, and S. Sorower
Dept. of Electrical and Computer Engineering, U. of Memphis, Memphis, TN, USA 38152;

ABSTRACT

This paper presents a simple prototype sparse detector imaging sensor built using sixteen off-the-shelf, retro-reflective, infrared-sensing elements placed at five-inch intervals in a vertical configuration. Profiling experiments for broad-scale classification of objects, such as humans, humans wearing large backpacks, and humans wearing small backpacks were conducted from data acquired from the sensor. Empirical analysis on models developed using fusion of various classifiers based on a diversity measure shows over ninety percent (90.07%) accuracy (using 10-fold cross validation) in categorizing sensed data into specific classes of interest, such as, humans wearing a large backpack. The results demonstrate that shadow images of sufficient resolution can be captured by the sensor such that broad-scale classification of objects is feasible. The sensor appears to be a low-cost alternative to traditional, high-resolution imaging sensors, and, after industrial packaging, it may be a good candidate for deployment in vast geographic regions in which low-cost, unattended ground sensors with highly accurate classification algorithms are needed.

Keywords: Sparse imaging sensor, object detection, border monitoring

1. INTRODUCTION

The overall goal of this research is to develop a network of low-cost sensors and intelligent signal processing algorithms that detect and provide a broad-scale classification of humans and vehicles, while ignoring animals, in a variety of applications, including border monitoring. Borders are often the point of entry for illegal aliens, smugglers, and terrorists and represent a significant challenge to national security. The United States' border with Canada and Mexico is approximately 12,000 km, with many remote and uninhabited sections. Automated monitoring of these areas is currently very difficult and is restricted to high-traffic areas; however, ubiquitous deployment of sensors to monitor the entire border is of extreme interest. For example, The Department of Homeland Security (DHS) is leading the development of technology for border security via the SBI.net project¹. This project is attempting to use advanced sensors, such as moving target indication (MTI) radar and thermal infrared cameras, in a network-centric environment, to perform detection, classification, and tracking of humans crossing the border. When fully developed, the estimated cost to deploy this technology along the entire U.S. border is approximately \$1million dollars per mile of border (approximately \$620K per km)².

Innovative use of alternative low-cost sensors and networking technology may provide a competing or complementary monitoring capability for borders and other application scenarios. For example, suppose a suite of low-cost sensors could be developed that provide a significant percentage of the functionalities offered by the technologies selected for the SBI.net project, but they could do so only for a coverage area of 10 m. If the cost of each sensor was \$100 and it costs an equal amount for installation, the total cost would be approximately \$20K per km of border. If each sensor had a 10-percent per year failure rate, there would be an additional recurring cost of \$2K per km. Even doubling these costs, there is still a substantial margin between the potential costs of the sensors envisioned as part of our work and the technologies currently under development for comprehensive electronic border security.

The remainder of this paper details the design and evaluation of one approach to a low-cost imaging sensor. It is proposed that such a sensor could be a component of a ubiquitous, low-cost sensor network for applications, such as border monitoring. Section 2 introduces our sparse detector imaging sensor prototype and an initial motivating application. Section 3 provides the details of the sensor developed in our laboratory and the details of the acquisition of data used for subsequent classification. Section 4 highlights an initial approach used for classifying the data from the sensor and summarizes some preliminary performance metrics. Section 5 offers conclusions and future directions.

*drussmnn@memphis.edu; phone 1 901 678-3253; fax 1 901 678-5469; www.eece.memphis.edu

2. SPARSE DETECTOR SENSORS

Sparse detector sensors are a low-cost alternative to traditional, high-resolution imaging sensors for object classification. Size, cost, and power restrictions preclude the use of traditional imagers in applications that require widespread deployment of the sensors in unattended ground applications. Classification of sparse detector sensor data is of extreme interest when building inexpensive, ground sensors for many government agencies. But robust classification is challenging due to the paucity of information that can be used to model various objects that may be sensed.

The design of an unattended ground sparse detector imaging sensor for broad-scale classification has been prototyped in our laboratory. This prototype sensor is being initially designed and evaluated in response to the need to monitor trails that provide routes for drug smuggler traffic, as well as other applications in which broad-scale classification from data acquired from unattended ground sensors is of high interest. Unattended ground sensors that can reliably distinguish between humans and animals are critically needed. For example, it is typical for drug smugglers on foot to use large backpacks to transport marijuana weighing up to 100 lbs along known trails across the border between the U.S. and Mexico. Smugglers often travel in large groups and the border patrol has inadequate personnel to monitor these vast geographic regions. Therefore, a high degree of confidence in classification algorithms is needed to provide notification when objects of interest are detected to allow authorities to assemble adequate personnel to intercept and apprehend the smugglers along known points on the trails. Moreover, numerous and inexpensive unattended ground sensors are needed for placement at several locations and these sensors should be resilient to false alarms as there is inadequate capacity among the authorities for reacting to false alarms.

In our initial application domain, such trails are abundant, but most of the routes are known. However, the routes have many travelers that are not of interest, including animals and humans that do not fit the profile of traveling in groups with large backpacks. In typical deployments, a sensor would be placed near a trail having a width of approximately 3 feet. These sensors can be located next to trails where vegetation can be used to hide placement. The trails are located in areas that are considered open range. Wild horses, cattle, deer, large cats, dogs, rabbits, and pigs are just a few of the animals that use the same trails as humans moving through the area.

3. SPARSE DETECTOR IMAGING SENSOR

3.1 Sensing elements

The CX-RVMS retro-reflective infrared sensor³ as shown in Fig. 1, was used as the principal sensing element to construct a prototype sparse detector sensor to obtain signature data in a laboratory environment for further analysis. The CX-RVMS was selected due to its suitability for both lab and controlled-field testing. It has environmental reliability, which includes BSi IP67 waterproof construction, and it is vibration resistant with its interior fully filled with resin. Moreover, this sensing element has a 1 ms or less response time, 5 m sensing range, consumes less than 40 mA, and can operate from -25 to 60 °C. An alternative sensing element would be required for wide-scale deployment; however, the CX-RVMS served our purposes for initial development and to acquire signature data.

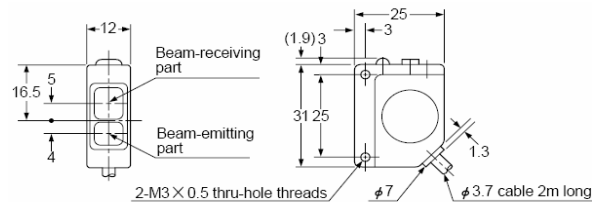


Fig. 1. CX-RVMS retro-reflective photoelectric sensing element (all units mm)³.

3.2 Sensor assembly

The prototype sparse detector sensor was assembled by placing 16 CX-RVMS retro-reflective sensing elements at 5-inch intervals in a vertical configuration. Each sensing element can be regarded as an optical trip wire and was attached

to a supporting pole with reflectors mounted on an opposing pole to provide the break-beam curtain. Each CX-RVMS sensing element was interfaced to a USB-DIO-32 digital input board using a simple voltage divider breadboard circuit to provide the required 0 to 5V output. The input board was subsequently connected to a host computer via a USB interface. Fig. 2 (left panel) illustrates the configuration in which sensing element B0 denotes the first optical trip wire placed at 5 inches off the ground through sensing element C7, which is placed at approximately 80 inches off the ground. Fig. 2 (right panel) is a photograph of the pole with the sensing elements interfaced to a host computer in the laboratory environment.

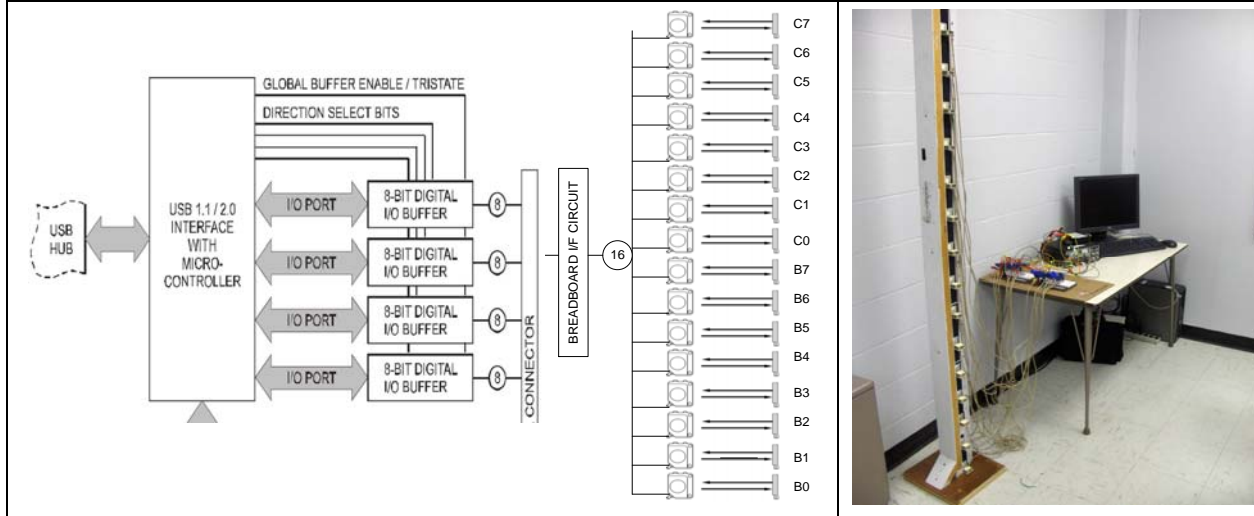


Fig. 2. Prototype sparse detector sensor.

3.3 Data acquisition

The driver software for the digital input board interfaced to the sensor uses a 32-bit DLL compatible with any Windows programming language. A C/C++ program was developed to acquire signature data assuming a single object traverses through the sensor’s optical curtain for initial testing purposes. Therefore, data acquisition started when a break-beam event first occurred and continued at a sampling rate of approximately 10ms until none of the sensing elements detected a break in their beam.

The C/C++ program writes a single ASCII file for each object detection event as a string of 1s and 0s corresponding to no-break and break, respectively, for sensing element B0 through C7 as the object passes through the sensor’s optical curtain. Each of the 16 sensing elements is sampled in parallel, which provides data for a ‘16 x I ’ matrix. Variable ‘ I ’ is the number of samples taken for each sensing element and will be constant within a single file; that is, the same number of samples will be taken for sensing elements B0 through C7 for a given object’s data acquisition. However, ‘ I ’ will vary among files as it depends on the specific object and the interval of time in which at least one sensing element detected a break-beam event, that is, ‘ I ’ depends on the speed, size, configuration, and other variables of the object.

Fig. 3 (right panel) is a visualization of the optical trip wires for a typical break-beam pattern for a human with a small backpack and was produced using the Visualization Toolkit (VTK)⁴ from data acquired from the sensor. For initial testing, a total of 272 data sets were acquired from the sensor. Fig. 4 represents the creation of a simple image using a MATLAB program from the break-beam patterns for a variety of objects including: a) a human with a small backpack, b) a human without backpack, c) a human with a large backpack, d) a small dog, e) a human walking a small dog; and f) an office chair. Recall that these images were produced from 16 detectors placed at 5-inch intervals.

In the case of multiple objects traversing through the sensor as in Fig. 4 e), our data acquisition program required a continuous break of at least one optical trip wire to acquire the data for both objects in one file. Multiple objects as in Fig. 4 e) were not used as data sets for the classification algorithms reported later in this paper. A number of samples were acquired for each object type consisting of different strides, postures, orientations, speed, etc.

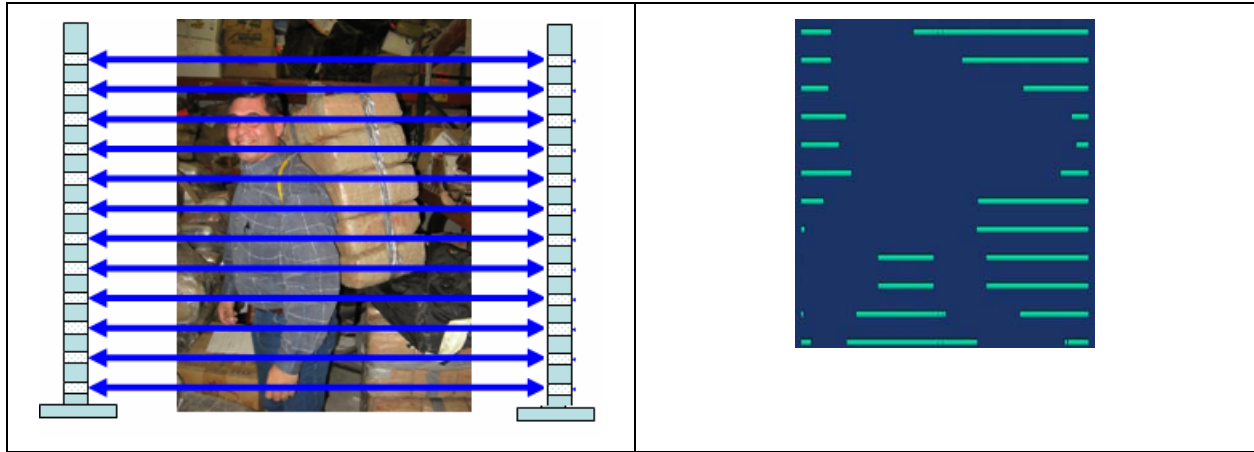


Fig. 3. Optical trip wire break-beam pattern.

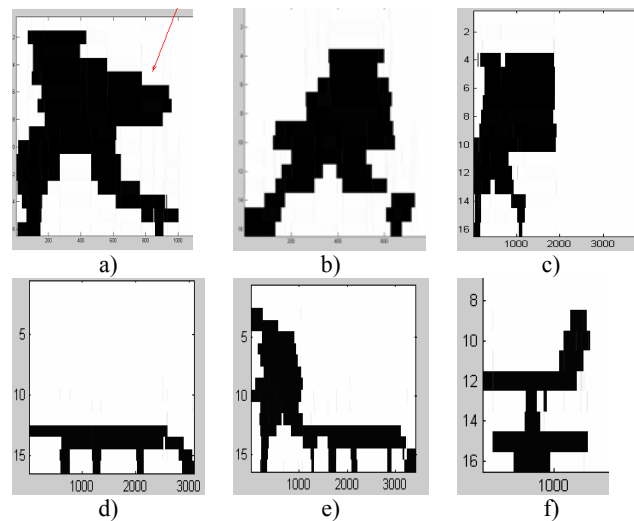


Fig. 4. Images created from break-beam patterns acquired from the sparse detector imaging sensor: a) human with small backpack, b) human without backpack, c) human with large backpack, d) small dog, e) human walking a small dog, and f) office chair.

3.4 Other deployment issues

To accommodate ubiquitous deployment, data communication protocols must be carefully considered. Minimizing wireless communications saves both bandwidth and power. The beam-break patterns of each sensor are highly compressible so that a relatively sparse representation could be transmitted over a wireless communication channel each time the sensor is activated. Alternatively, a classification algorithm could be executed on an embedded computer and a high-level report of the classification output sent over a wireless link. For this to be feasible, the algorithm must be computationally efficient and easily implemented on a low-cost, low-power computing platform.

Power requirements are also a major concern as long battery life is highly desirable. The choice of detectors affects power consumption along with the choice of data communication protocols. Transmission of high-level classifications consumes less power than the transmission of profile data.

Concealment of the device is also desired. To be effective, a smuggler or person crossing a border illegally must not be aware of the sensors' location. Fig. 5 demonstrates a simple means of camouflaging a notional packaging of a profiling

sensor using paint. Another means of hiding the device is to distribute the sensors horizontally along the path. Each beam-break device would operate at both a different height and a different distance along the trail. By distributing the sensors, the profiling sensor is less conspicuous. By keeping track of the location of each sensor, the profile can be constructed by synchronizing the data from the distributed beam-break sensors. While the sensing elements described in this research are an active near-infrared emitter/detector, it would be possible to construct a profiling sensor from passive long-wave infrared detectors. These types of sensors are often used in motion detectors for outdoor lighting. Lenses could be used to provide a sufficiently narrow field-of-view to capture one element of a profile. This type of detector has the advantage of not requiring a reflector to operate. In addition, it is undetectable using night vision imaging devices.



Fig. 5. Example camouflage of sparse detector sensor: a) Profiling sensor realized with sensing elements in PVC pipe and mounted on a tree b) Profiling sensor realized with sensing elements in PVC pipe painted to blend in with a tree.

4. SPARSE DETECTOR SENSOR DATA CLASSIFICATION

To address the classification of data obtained from our sparse detector sensor, this section summarizes preliminary empirical analysis using the fusion of classifiers to develop robust models of objects of interest. The classification of sparse detector sensor data summarized in this paper consists of two main modules: i) data representation, and ii) modeling objects of interest. The following subsections summarize the basic concepts and the classification techniques used in this paper for completeness before presenting their application on the sparse sensor data.

4.1 Data representation

The data representation module has two main components, namely, data processing and feature selection modules.

Data processing: The sensed data is divided into three categories based on the types of objects of interest (e.g., human with large backpack, human with small backpack, and human without backpack). For the experiments summarized in this paper, the number of samples obtained from the sensor in each category varies from one class to the other (e.g., person with backpack may have more samples than person without backpack).

Feature extraction: As previously described, each object sample is a $16 \times I$ matrix. Principle component analysis (PCA) is used to project the data onto a lower dimensional space to form feature vectors of equal length and to remove redundancies from the data. Each data sample is projected from $16 \times I$ to 16×10 and is used as a feature vector for modeling various objects. At least 85% of the energy of the signal is preserved in the transformation. Each projected sample is further arranged in lexicographical order to convert the 16×10 matrix into 1×160 vectors. The 160 dimensional feature vectors were used to model various objects of interest (e.g., human with backpack, etc.).

4.2 Classification

A diverse set of classifiers, such as individual classifiers (i.e., sequential minimal optimization, function-based classifier, etc.), ensemble classifiers (i.e., bagging, boosting, random forest, etc.), and fusion of multiple classifiers have been used for robust modeling of objects and their performance were compared. The WEKA toolbox⁵ was used for developing models of objects, such as human, human with large backpack, etc., using various classification algorithms. The fusion of classifiers was performed based on measures of diversity as opposed to random choices of classifiers. In-depth

performance evaluation of the various models was performed to show the utility of each model in classifying such sparse data and also to potentially identify a classifier that might be a candidate for implementation in a deployable version of the prototype sensor. The following subsections summarize the basic concepts and the utility of diversity measures and fusion techniques and performance evaluation methods used in this paper for completeness before presenting their application on the sparse sensor data.

4.3 Measure of diversity among classifiers

Different classifiers are designed based on various prior assumptions and following different mathematical procedures to model an underlying function. It is quite possible for different classifiers to produce different results (i.e., make mistakes at different data points) on the same data set even though the accuracy for different classifiers is the same. It is expected that fusion of classifiers can produce better models compared to the individual classifiers if the fused classifiers are different (i.e., independent, orthogonal, and complementary). The theoretical framework leads to diverse multiple classifier system architectures. For a set of classifiers, a good choice of diversity measures is based on entropy (E) and is defined as follows⁶:

$$E = \frac{1}{N} \sum_{j=1}^N \frac{1}{L - \left\lfloor \frac{L}{2} \right\rfloor - 1} \min \left\{ \sum_{i=1}^L y_{j,i}, L - \sum_{i=1}^L y_{j,i} \right\}$$

in which L = total number of classifiers, N = number of data instances, $y_{j,i} = 1$ if i^{th} classifier correctly classifies j^{th} data point and 0 otherwise. The higher the value of E indicates increased diversity, with 1 being the highest possible diversity and 0 indicating no diversity.

There are a few other diversity measures for a set of classifiers which are based on variance measure, such as Kohavi-Wolpart (KW) variance⁶. KW measure is defined as follows:

$$KW = \frac{1}{NL^2} \sum_{j=1}^N l(z_j)(L - l(z_j))$$

in which $l(z_j) = \sum_{i=1}^L y_{j,i}$ and the other symbols have the meaning as defined earlier. This expression calculates the averaged variance of the Bernoulli variable y with values 0 for incorrect and 1 for correct classification of each object. The choice of classifiers was made based on both measures of diversity to improve the accuracy and robustness.

4.4 Classifier fusion

Given a training data set D (drawn from a common instance space X) for supervised learning and a collection of inductive learning algorithms, and hypothesis languages (H); find a new prediction algorithm (not necessarily $\in H$) for $x \in X$ that combines outputs from the collection of classifiers. Hypotheses are produced by applying inducers to $s(D)$ where $s: X \text{ vector} \rightarrow X' \text{ vector}$ (sampling, partitioning, etc.)⁷. There are number of ways to combine the output of the base classifiers. One of the most popular and simplest ways to combine the output of multiple classifiers is within a voting framework. Let $C^1 \dots C^N$ be the set of classifiers that are induced by training N different learning algorithms $L^1 \dots L^N$ on a data set D consisting of feature vectors. To classify a new instance at runtime, the classifiers are queried for a class value and the class with the highest count is finally selected. This scheme is known as majority (or plurality) voting. Variations include weighted majority voting and voting using class probability distributions. In the former approach, each classifier's vote is weighted by its accuracy, as measured by either using a holdout data set or the entire training data set by cross-validation. Generalized stacking⁸ is also used to combine classifiers which use more sophisticated algorithms to learn how to combine the outputs of a set of classifiers. Various versions of stacking (e.g., radial basis function, etc.) and versions of voting (e.g., weighted majority voting, etc.) from WEKA were used for the fusion of classifiers in this work. Subsequent subsections summarize the weighted majority voting and stacking for completeness.

4.5 Weighted majority voting fusion

The key idea in weighted majority voting fusion is to collect votes from a pool of prediction algorithms (classifiers) for each training example. The weight associated with each algorithm that guessed wrong is decreased by a multiplicative factor. The combiner predicts the weighted majority label. Each classifier h_i maps from $x \in X$ to $h_i(x)$ resulting in a prediction for a set of legal class labels. The key goal is to improve the training set accuracy by combining weak

classifiers and by bounding the number of mistakes in terms of a minimum made by any one algorithm⁹. The pseudo code below succinctly summarizes the weighted-fusion process.

Weighted-Majority Fusion (D, L)¹⁰:

```

1.   $n \leftarrow L.size$  // number of inducers in pool
2.   $m \leftarrow D.size$  // number of examples  $\langle x \equiv D[j], c(x) \rangle$ 
3.  FOR  $i \leftarrow 1$  TO  $n$  DO
     $P[i] \leftarrow L[i].Train-Inducer(D)$  //  $P[i]$ :  $i$ th prediction algorithm
     $w_i \leftarrow 1$  // initial weight
4.  FOR  $j \leftarrow 1$  TO  $m$  DO // compute WM label
     $q_0 \leftarrow 0, q_1 \leftarrow 0$ 
5.  FOR  $i \leftarrow 1$  TO  $n$  DO
6.  IF  $P[i](D[j]) = 0$  THEN  $q_0 \leftarrow q_0 + w_i$  // vote for 0 (-)
7.  IF  $P[i](D[j]) = 1$  THEN  $q_1 \leftarrow q_1 + w_i$  // else vote for 1 (+)
8.   $Prediction[i][j] \leftarrow (q_0 > q_1) ? 0 : ((q_0 = q_1) ? Random(0, 1) : 1)$ 
9.  IF  $Prediction[i][j] \neq D[j].target$  THEN //  $c(x) \equiv D[j].target$ 
10.  $w_i \leftarrow \beta w_i$  //  $\beta < 1$  (i.e., penalize)
11. RETURN  $Make-Predictor(w, P)$ 

```

4.6 Combination using stack generalization

Wolpert introduced a novel approach for combining multiple classifiers known as stacked generalization or stacking⁸. The key idea is to learn a meta-level (or level-1) classifier based on the output of base-level (or level-0) classifiers estimated via cross-validation as follows. Define D , a data set consisting of feature vectors, also referred to as level-0 data, and $L^1 \dots L^N$ a set of N different learning algorithms. During a K -fold cross-validation process, D is randomly split into K disjoint parts $D^1 \dots D^K$ of almost equal size. At each k^{th} fold, $k = 1 \dots K$, the learning algorithms $L^1 \dots L^N$ are applied to the training part $D \setminus D^k$ and the induced classifiers $C^1(k) \dots C^N(k)$ are applied to the test part D^k . The concatenated predictions of the induced classifiers on each feature vector x_i in D^k , together with the original class value $y_i(x^i)$, form a new set MD^k of meta-level vectors. At the end of the entire cross-validation process, the union of MD^k , $k = 1 \dots K$ constitutes the full meta-level data set, also referred to as level-1 data, which is used for applying a learning algorithm and inducing the meta-level classifier. The learning algorithm, which is employed at the meta-level, could be one of the $L^1 \dots L^N$ or a different one. Finally, the $L^1 \dots L^N$ learning algorithms are applied to the entire data set D inducing the final base-level classifiers $C^1 \dots C^N$ to be used at runtime. To classify a new instance, the concatenated predictions of all base-level classifiers $C^1 \dots C^N$ form a meta-level vector that is assigned a class value by the meta-level classifier C^M . The procedure for the stack generalization is summarized as follows:

Stack Generalization⁸:

Algorithm Combiner-Stacked-Gen ($D, L, k, n, m', Levels$)

```

1.  Divide  $D$  into  $k$  segments,  $S[1], S[2], \dots, S[k]$  // Assert  $D.size = m$ 
2.  FOR  $i \leftarrow 1$  TO  $k$  DO
    a.  $Validation-Set \leftarrow S[i]$  //  $m/k$  examples
3.  FOR  $j \leftarrow 1$  TO  $n$  DO
     $Train-Set[j] \leftarrow Sample-With-Replacement(D \sim S[i], m')$  //  $m - m/k$  examples
    IF  $Levels > 1$  THEN
     $P[j] \leftarrow Combiner-Stacked-Gen(Train-Set[j], L, k, n, m', Levels - 1)$ 
    ELSE // Base case: 1 level
     $P[j] \leftarrow L[j].Train-Inducer(Train-Set[j])$ 
4.   $Combiner \leftarrow L[0].Train-Inducer(Validation-Set.targets, Apply-Each(P, Validation-Set.inputs))$ 
5.   $Predictor \leftarrow Make-Predictor(Combiner, P)$ 
RETURN  $Predictor$ 

```

All of the classifiers applied to the sparse detector sensor data were evaluated using N -fold cross validation and using various performance metrics, such as receiver operating characteristic (ROC) curves¹¹⁻¹². Classifiers produce a confusion/contingency matrix, which show four entities: True positive (TP), True negative (TN), False positive (FP), and

False negative (FN). Based on these quantities, a number of performance indices, such as precision ($P = TP/(TP + FP)$), recall ($R = TP/(TP + FN)$), True positive rate ($TPR = TP/(TP + FN)$), and False positive rate ($FPR = FP/(FP + TN)$) can be defined. In addition, F-measure is defined based on the harmonic mean of precision and recall F-measure: $F = 2 * Recall * Precision / (Recall + Precision)$. A thorough performance of each model was computed and compared on classifying the sparse sensor data.

4.7 Classification results

A total of 272 data samples were collected using the setup described in Section 3. The labels and number of samples for each class were as follows: Class 1: Human with large backpack (100); Class 2: Human with no backpack (112) and Class 3: Human with small backpack (60).

The following abbreviations were used to summarize the results in tabular form: NB–Naïve Bayes; BN–BayesNetwork; SMO–Sequential Minimal Optimization; REPT–REPTree; J48–J48 Decision Tree (C4.5); MP–Multilayer Perceptron; RDR–Ridor, MCC–Multilayer Classifier; LB–LogitBoost; BG–Bagging; RF–Random Forest; StkCRBF–LBBGRF StackingC with RBF Network using LB, BG, and RF; Vote–AVG–BGLBRF Vote with Average Probability using RF, BG, and LB; Vote–PRD–BGLBRF Vote with Product of Probabilities using BG, LB, and RF; StkCRBF–LBBGBN StackingC with RBF Network using LB, BG, and BN; StkCRBF–LBBG StackingC with RBF Network using LB and BG; MAE–Mean absolute error; RMSE–Root mean squared error; RAE–Relative absolute error; and RRSE–Root relative squared error.

Table 1 summarizes the accuracy (using 10-fold cross validation) and errors of various classifiers applied to the sparse sensor data. Among the individual classifiers, the Ripple-Down Rules classifier (RIDOR) recorded the highest accuracy (86.40%). The ensemble classifiers improved the overall accuracy. Both the bagging- and boosting-based methods recorded similar accuracy (87.87%). The fusion of classifiers scheme recorded (90.07%) accuracy, marginally improving the performance. The following observations can be made from Table 1. It is easy to note that ensemble classifiers, in general, performed better than the individual classifiers and the fusion of classifiers marginally improved the accuracy over ensemble classifiers. The fusion of classifiers recorded the lowest Mean Absolute Error (MAE), Root Mean Squared (RMS) error, Relative Absolute Error (RAE) and Root Relative Squared Error (RRSE). For those classifiers that were compared, lower error measures implied better performance.

Table 1. Classification performance of classifiers for 3 classes with 10-fold cross-validation.

Classifiers		Correctness		Error Measures			
		Accuracy	Kappa	MAE	RMSE	RAE	RRSE
Individual Classifiers	NB	83.09%	0.7415	0.1126	0.3263	26.10%	70.27%
	BN	85.29%	0.7771	0.0984	0.2914	22.81%	62.76%
	SMO	78.68%	0.665	0.2761	0.3562	64.03%	76.73%
	REPT	83.09%	0.7387	0.1457	0.3075	33.78%	66.23%
	J48	84.19%	0.7552	0.1072	0.3206	24.85%	69.05%
	MP	79.41%	0.6755	0.1471	0.3284	34.11%	70.73%
	RDR	86.40%	0.788	0.0907	0.3011	21.03%	64.86%
Ensemble Classifiers	MCC	73.53%	0.590	0.3414	0.3920	79.16%	84.43%
	LB	87.87%	0.8097	0.1154	0.2473	26.77%	53.28%
	BG	87.87%	0.8097	0.1335	0.2487	30.95%	53.57%
	RF	86.40%	0.7863	0.1860	0.2759	43.14%	59.43%
Fusion of Classifiers	StkCRBF-LBBGRF	90.07%	0.8444	0.1185	0.2322	27.48%	50.02%
	Vote-AVG-BGLBRF	89.34%	0.8333	0.145	0.2365	33.62%	50.93%
	Vote-PRD-BGLBRF	90.07%	0.8448	0.0754	0.2381	17.47%	51.29%
	StkCRBF-LBBGBN	89.71%	0.8401	0.1225	0.2333	28.40%	50.26%
	StkCRBF-LBBG	89.71%	0.8388	0.1271	0.2367	29.47%	50.99%

The choice of which classifiers to fuse was made based on their diversity scores. Table 2 summarizes these scores for various combinations of classifiers. From Table 2 it is easy to see that the second group: Bagging (BG), LogitBoost (LB), and Random Forest (RF), has a better diversity measure value relative to the other groups. This justifies the choices of classifiers for the fusion reported in Table 1. Expectedly, the fusion of those classifiers yields the highest accuracy and better performance measures (see Table 3). To further illustrate the utility of the diversity measure in fusion, a simple pair-wise measure of diversity was also used. It was found that Group 4 has the lowest KW value (pair-wise entropy is undefined). The classification accuracy for the Group 4 set of classifiers is marginally inferior compared to Group 1, but has a lower computational cost than Group 1. This provides a trade-off between accuracy of the model and computational time, which may be an issue when considering deployment of the system.

Table 2. Diversity among classifiers and fused performance (3 classes).

N	Classifiers	Accuracy	Diversity		Fusion Accuracy (%)
			Entropy	KW	
1	Bagging (BG) LogitBoost (LB) Random Forest (RF)	87.87% 87.87% 86.40%	0.143	0.032	90.07%
2	BayesNet (BN) LogitBoost (LB) Bagging (BG)	85.29% 87.87% 87.87%	0.214	0.048	89.71%
3	Multilayer Perceptron (MP) LogitBoost (LB) Bagging (BG)	79.41% 87.87% 87.87%	0.179	0.040	88.60%
4	Tree J48 (J48) LogitBoost (LB) Bagging (BG)	84.19% 87.87% 87.87%	0.107	0.024	85.29%
5	Bagging (BG) LogitBoost (LB)	87.87% 87.87%	-	0.027	89.71%
6	LogitBoost (LB) Random Forest (RF)	87.87% 86.40%	-	0.018	88.97%

Table 3. Performance analysis of different fusion classifiers.

Classifiers		CLASS	TPR	FPR	P	R	F	ROC-A
Fusion of Classifiers	StkCRBF-LBBGRF	CLASS1	0.98	0.052	0.916	0.98	0.947	0.989
		CLASS2	0.955	0.075	0.899	0.955	0.926	0.963
		CLASS3	0.667	0.028	0.87	0.667	0.755	0.902
	Vote-AVG-BGLBRF	CLASS1	0.97	0.047	0.924	0.97	0.946	0.994
		CLASS2	0.946	0.075	0.898	0.946	0.922	0.984
		CLASS3	0.667	0.042	0.816	0.667	0.734	0.94
	Vote-PRD-BGLBRF	CLASS1	0.97	0.041	0.933	0.97	0.951	0.994
		CLASS2	0.955	0.075	0.899	0.955	0.926	0.98
		CLASS3	0.683	0.038	0.837	0.683	0.752	0.94
	StkCRBF-LBBGBN	CLASS1	0.98	0.041	0.933	0.98	0.956	0.99
		CLASS2	0.911	0.063	0.911	0.911	0.911	0.963
		CLASS3	0.733	0.052	0.8	0.733	0.765	0.939
	StkCRBF-LBBG	CLASS1	0.98	0.052	0.916	0.98	0.947	0.985
		CLASS2	0.946	0.075	0.898	0.946	0.922	0.955
		CLASS3	0.667	0.033	0.851	0.667	0.748	0.896

While the accuracy and MAE, RMS, RAE, and RRSE error provides some indication of the performance of the models, to have an in-depth performance evaluation, it is imperative to understand the utility and efficacy of the models. Table 3, summarizes the in-depth performance analysis of various combination of the fused classifiers. The performance analysis of individual and ensemble classifiers is not reported here as they have relatively poor performance compared to the fused classifiers. However, in-depth performance analysis using various fusion schemes is reported in Table 3.

The following observations are made from Tables 3. It is easy to notice the low FPR and high TPR, P, R, and F for all fused classifiers in modeling Class 1 and Class 2, but the same performance measures are relatively inferior for Class 3. Based on these performance measures, it can be inferred that the proposed methods were capable of modeling Class 1 and Class 2 quite effectively, and Class 3 would require further data collection and remodeling before practical deployment. It is also noted that while both fusion schemes have comparable performance, the fusion using stack generalization yields the best accuracy, while the fusion using the average voting yields the better ROC area. It should also be noted that the fusion of classifiers, based on diversity, has improved the accuracy by approximately 2%-3% compared to ensemble and individual classifiers, but it has significantly improved performance metrics across the board. To further analyze the results, the confusion matrix for the best individual, ensemble, and fused classifiers is presented in Table 4. From the confusion matrix, it is observed that the person with a small backpack occasionally gets confused with a person with no backpack or a person with a large backpack. This is also consistent with common sense observation given the ad-hoc definitions of the classes.

Table 4. Confusion matrix for the best individual classifier (RIDOR), ensemble classifier (Bagging), and fusion of classifiers (StkCLR-LBBGRF).

Classified as →	Ridor (RDR)			LogitBoost (LB)			StkCRBF-LBBGRF		
	a	b	c	a	b	c	a	b	c
a = Class 1	95	1	4	96	0	4	98	0	2
b = Class 2	2	101	9	1	106	5	1	107	4
c = Class 3	7	14	39	7	16	37	8	12	40

5. CONCLUSIONS AND FUTURE DIRECTIONS

A simple prototype imaging sensor has been presented to show the feasibility of using a sixteen detector array implemented as retro-reflective sensing elements placed in a vertical configuration for discriminating among humans, humans wearing large backpacks, and humans wearing small backpacks. Such an approach to a sparse detector sensor may be critical for wide-scale deployments in which the cost of the sensor is crucial. An initial set of algorithms from the WEKA toolbox were evaluated along several dimensions with respect to their ability to classify the data acquired from the sensor. Empirical analysis shows over ninety percent (90.07%) accuracy (using 10-fold cross validation) in categorizing sensed data into specific classes of interest. Although not included as part of the results summarized in this paper, the sensor and associated algorithms appear to be resilient to false alarms induced from animal signatures; however, more detailed analysis is required to determine ‘false alarm’ probabilities.

Future directions are being pursued along a variety of focus areas, including acquiring more signature data from the prototype sensor for a wider variety of objects typically found in open range, such as wild horses, cattle, deer, large cats, dogs, rabbits, and pigs. Also, signature data for a variety of vehicles, such as SUVs, ATVs, light trucks, cars, motorcycles, bicycles, etc., is of interest along with further analysis of algorithms to classify the signature data.

It may be required to execute the detection and classification algorithms embedded within the sensor using a light-weight, low-cost computing platform. A variety of algorithms will be assessed with respect to their accuracy and their ability to successfully function on embedded platforms. Such an approach will reduce communications traffic, which is often a major source of power consumption in ubiquitous sensing environments. Also, minimizing network bandwidth by transmitting only high-level classification results, as opposed to low-level profile data is desirable.

A power consumption analysis and reduction effort is needed prior to deployment, as well as developing techniques for camouflage or covert placement. Development of a second version of the prototype sensor, which will be more durable for further field testing in typical application scenarios, is planned. Ultimately, the realization of the prototype sparse

detector sensor concept as a 'sensor on a stick' with a self-contained power supply and an embedded system for light-weight computation and communications within a network-centric, service-oriented environment is of high interest.

ACKNOWLEDGEMENT

Funding for this work was provided in part by agreement W911NF-05-2-0019 between the U. of Memphis and the U.S. Army's Research Lab (ARL), as well as funds from the Herff College of Engineering at the U. of Memphis. This paper does not necessarily represent the position of the U.S. Government.

REFERENCES

- ¹ W. Dizard, "DHS unveils massive, fast-track border project," *Government Computer News*, January 26, 2006.
- ² C. Strohm, "Border tech program is plagued by early setbacks," *Government Executive*, June 27, 2007.
- ³ Anon., "CX-RVM5/D100/ND300R data sheet," Ramco Innovations, W. Des Moines, IA, 2007.
- ⁴ W. Schroeder, L.S. Avila, and W. Hoffman, "Visualizing with VTK: a tutorial," *IEEE Computer Graphics and Applications*, 20-27, September/October, 2000.
- ⁵ I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques*, 2nd ed., San Francisco: Morgan Kaufmann, 2005.
- ⁶ L.I. Kuncheva and C. J. Whitaker, "Ten measures of diversity in classifier ensembles: limits for two classifiers," *Proc. of IEEE Workshop on Intelligent Sensor Processing*, Vol. 1, Birmingham, UK, 2001, pp. 10-16.
- ⁷ T.G. Dietterich, "Machine learning research: four current directions," *AI Magazine*, Vol. 18(4), pp. 97-136, 1997.
- ⁸ D. H. Wolpert, *Stacked generalization*, Vol. 5, Pergamon Press, 1992.
- ⁹ G. Sigletos, G. Paliouras, and C.D. Spyropoulos, "Combining information extraction systems using voting and stacked generalization," *Journal of Machine Learning Research*, Vol. 6, pp.1751-1782, 2005.
- ¹⁰ W. H. Hsu, "Combining classifiers: weighted majority, bagging, and stacking," 2001, <http://www.kddresearch.org/Courses/Fall-2001/CIS732/Lectures/Lecture-21-20011106.pdf>.
- ¹¹ T. M. Mitchell, *Machine Learning*, Singapore: McGraw-Hill, 1997.
- ¹² N. M. Adams and D. J. Hand, "Improving the practice of classifier performance assessment," *Neural Computation*, Vol. 12, pp. 305-312, 2000.