Received: 13 March 2018 | Revised: 1 September 2018 | Accepted: 23 October 2018

DOI: 10.1002/rob.21848

REGULAR ARTICLE

WILEY

# An image recognition system aimed at search activities using cyber search and rescue dogs

Solvi Arnold[1] | Kazunori Ohno[2] | Ryunosuke Hamada[2] | Kimitoshi Yamazaki[1]

[1]Department of Mechanical Systems Engineering, Shinshu University, Nagano, Japan

[2]New Industry Creation Hatchery Center, Tohoku University, Sendai, Japan

**Correspondence**
Solvi Arnold, Department of Mechanical Systems Engineering, Shinshu University, 4-17-1 Wakasato, Nagano city, Nagano, 380-8553 Japan.
Email: s_arnold@shinshu-u.ac.jp

**Funding information**
Council for Science, Technology and Innovation (Cabinet Office, Government of Japan)

## Abstract

Disaster response presents major challenges for robotics and computer vision alike. The Cyber-Enhanced Canine Suit is a suit equipped with a camera, Global Navigation Satellite System (GNSS), and various other sensors, to be worn by search and rescue (SAR) dogs for the purpose of enhancing SAR dog operations. This paper presents an image recognition system for use in disaster scenarios and its integration with the Cyber-Enhanced Canine Suit platform. The system's intended use is to spot personal items of missing individuals or other visual clues in video streams from various disaster response platforms. The system facilitates quick learning of targets from limited data and makes providing that data quick and easy. It also provides backtrack recognition functionality, to rapidly find novel targets in the seen footage. We evaluated the recognition system on footage gathered in the field, obtaining promising results. Integrated with the Cyber-Enhanced Canine Suit, the system can automatically plot detections of search targets onto a map display, to provide operators with a quick overview of what was seen where.

**KEYWORDS**
emergency response, learning, perception

## 1 | INTRODUCTION

To minimise human suffering and loss of life after natural disasters, such as earthquakes, it is desirable to gather information on the affected environment and to perform search activity quickly. However, such environments are not always easily accessible for human search teams. Search activity plans should be implemented under various considerations, such as prevention of secondary accidents.

One solution is to use remote-operated entities. For example, when problems occurred in Fukushima nuclear power plant following the March 2011 Tohoku earthquake, a crawler-type remote-controlled robot was introduced in the nuclear power plant and played a role in grasping the internal situation (Nagatani et al., 2013). Recent years have seen the development of numerous remote-operated robots for use in places that are hazardous or inaccessible for humans (Kalouche, Rollinson, & Choset, 2015; Lee, Kang, Kim, & Park, 2004; Murphy, Kravitz, Stover, & Shoureshi, 2009).

Generally, the movement speed of current remote-controlled ground robots is slower than that of a human being. Therefore, while search activities using such robots have the above-mentioned advantages, one must assume that long operation times are required. The lack of speed has its causes in both hardware and human control. Designing hardware that can move around swiftly on a variety of rough terrains, such as rubble piles, remains a hard challenge. Remotely controlling a robot on rough terrains is a complex and challenging task, which precludes high operation speeds unless terrain recognition and low-level control are automated to a high degree.

Meanwhile, there is cooperation with search and rescue (SAR) dogs as a method of searching at the disaster site. Dogs have higher mobility than humans and are capable of moving around rough terrain with only high-level instruction from their handler. Furthermore, dogs have a sharp sense of smell that is invaluable in locating victims. Therefore, prompt rescue activities can be expected.

  

Combining the high mobility and olfactory abilities of SAR dogs with sensor and tracking technology could produce a highly effective disaster response platform. To this end, we have been developing the Cyber-Enhanced Canine Suit, equipped with a variety of sensors, to be worn by SAR dogs (Sakaguchi, Ohno, Takeuchi, & Tadokoro, 2013). The present study presents an image recognition system integrated with the suit, using a subset of the equipment included in the suit [camera, Global Navigation Satellite System (GNSS), radio equipment]. The contributions of this study are as follows:

1. We have constructed an image recognition system for use in search activities. The core functionalities of the system are image segmentation and classification. What is noteworthy is that we can register classification targets online, to respond flexibly at the disaster site.
2. To support an intuitive understanding of when what was observed where, we implemented a Graphic User Interface (GUI) that combines recognition results with position measurements from the GNSS unit and displays them spatially.
3. We propose a mechanism to perform recognition by tracing back through previously observed image data. This makes it possible to respond quickly when new search targets are added in the middle of the search activity.
4. Through experiments at sites simulating disaster-affected environments, we have evaluated the system's recognition capabilities and identified open issues for future development.

The structure of this paper is as follows. In Section 2, we will summarise the benefits of using SAR dogs and discuss related work. Section 3 explains the concepts of the image recognition system and explains the differences with conventional image recognition methods. Section 4 explains the two recognition modalities in detail. Section 5 discusses system operation. Section 6 presents our evaluation experiments on data gathered in a rubble field. Section 7 discusses integration with the cyber-enhanced canine suit and our field experiments with the integrated system. Section 8 summarises the paper and outlines several avenues of future study.

## 2 | PLATFORM: ROBOTISED SEARCH SYSTEM WITH SAR DOGS

### 2.1 | Present use of dogs in search activities

The use of dogs to assist in search operations has a long history. Currently, search dogs are trained for a variety of purposes, such as mountain SAR dogs, disaster relief dogs, and police dogs. Although we have no quantitative summary, even in the March 2011 Tohoku earthquake, many SAR dogs were deployed to search for victims buried in rubble.

To understand the characteristics of SAR dog operations, we consulted with the Japan Rescue Dog Association (JRDA), Japanese firefighters, and referred to the methods of the Swiss Disaster Dog Association (REDOG). In a typical mission, the rescue team first determines a set of initial search regions on basis of the characteristics of a search site, information from people familiar with the area, and any available clues, such as left behind items. Search is initiated in these regions. Subsequently, search regions are refined and narrowed down as new clues and insights come up during operation. The importance of visual clues in determining where to search is one of the motivating factors for integrating an image recognition system into the cyber-enhanced canine suit platform.

SAR dogs move around in rough rubble environments with excellent mobility. They discover victims with a keen sense of smell, which is said to be one hundred to one million times as sensitive as the human sense of smell, and bark to convey locations of interest to their handlers. At disaster sites, SAR dogs' activity time is in the order of several hours, including standby and movement. Search is conducted in sessions of about 10 min at a time, interspersed with breaks.

### 2.2 | Related work on disaster SAR dogs

Attempts to collect information on disaster sites using dogs equipped with sensors, such as cameras, have been reported in Ferworn et al. (2006) and Ferworn (2009). A recent study proposed a system that facilitates understanding of the state of a damaged building, using imagery from a pair of cameras attached to the sides of the dog (Ferworn, Waismark, & Scanlan, 2015). Bozkurt et al. (2014) too used a dog-mounted camera to facilitate understanding of disaster situations and for prediction of dog behaviour.

It has been suggested that dogs and disaster response robots jointly conduct exploration. Interesting examples of "marsupial" cooperative operation of SAR dogs and disaster response robots are proposed in Tran, Ferworn, Gerdzhev, and Ostrom (2010) and Ferworn, Wright, Tran, Li, and Choset (2012), demonstrating that SAR dogs can be used to quickly deliver small robots to locations of interest in a disaster site.

A challenge in position tracking for SAR dogs is the unreliability of GNSS signals in certain types of disaster sites. Collapsed buildings cause disturbances in GNSS-based geolocation. This issue was addressed for the specific case of SAR dogs in Sakaguchi et al. (2013) by estimating the dog's trajectory using the inertial measurement unit (IMU) and fusing the trajectory estimates from GNSS and IMU using an extended Kalman filter. In the ongoing study, IMU-based position estimation is being extended for use in locations where GNSS signals may be unavailable for extended periods of time.

### 2.3 | The proposed search system with SAR dogs

The authors have been developing a cyber-enhanced canine suit to be worn by SAR dogs (Sakaguchi et al., 2013). This suit weighs less than 10% of the weight of the SAR dog and takes weight balance into account. Camera, microphone, inertial sensor (IMU), GNSS, and data recording/distribution device are embedded. Data obtained from these sensors during exploration activities can be recorded on a micro secure digital (SD) card. In addition, it is possible to present

information, such as the visual scene the dog is seeing and the sound heard, in real time, on a tablet personal computer (PC) in a remote location.

In search activities using SAR dogs, it is necessary to grasp the situation at the search site quickly. For this reason, it is desirable to provide operators at remote locations with live footage with minimal delay. Here, image quality and framerate should be sufficient for search activity. In this subsection, we explain the structure of the search system developed to meet these requirements as well as possible. System specifications were set in cooperation with the JRDA. JRDA employs numerous active SAR dogs and has experience working in disaster sites.

Figure 1 shows an overview of the search system we constructed. In the search using the cyber-enhanced canine suit, video and audio of the surrounding environment are acquired by a video camera installed in the suit. Video and audio are distributed using a mobile phone line. Also, the positional change of the dog is measured as the latitude/longitude value given by the GNSS or the inertial sensor. Coordinate measurements are uploaded to a NoSQL (i.e. non-relational) database, provided via Amazon Web Services' (AWS) DynamoDB, via the mobile phone network. These data are received on tablet terminals, smartphones, and so forth. They can be viewed with a web browser-based GUI. In this way, information can be shared in real time by multiple people at the rescue site.

Image transfer is handled as follows. For the video, we confirmed in discussion with the JRDA that a Video Graphics Array (VGA) (640 × 480) resolution and 1–5 frames per second (fps) framerate are sufficient. A video camera meeting these specifications (Panasonic HX-A1H, Panasonic, Kadoma, Japan) is placed on the suit, located on the shoulder of the dog (Figure 2), and a mechanism is adopted to compress and deliver the picture taken there. Video is captured by a Single Board Computer (Raspberry Pi 2 Model B, Raspberry Pi Foundation, Cambridge, UK) on the suit and compressed in real time in H.264 format using a built-in hardware encoder. The compressed video is uploaded to the video distribution server on the 3G/4G mobile line using a mobile wireless fidelity (WiFi) router (Aterm MR 04 LN, NEC Corporation, Tokyo, Japan). For video uploading and stream serving, we adopted Web Real-Time Communication (WebRTC), which enables delivery with low delay. Image processing is performed remotely on the video stream, enabling us to implement image processing on high-performance hardware. This allows for computationally intensive video processing by means of graphics processing unit (GPU) and multithreaded programming.

## 3 | IMAGE RECOGNITION SYSTEM

Bringing robotic technology into the field of disaster response provides opportunities for supporting response teams' understanding of the state and content of disaster sites. Almost if not all robotic disaster response platforms are fitted with at least one camera. Every robot moving in the field contributes potentially valuable visual data. Cyber-enhanced rescue dogs are no exception in this regard.

However, having a lot of data does not automatically give responders a good understanding of the site. By processing data into intuitive overviews and identifying locations of high interest, the cost (in terms of time and human resources) of data interpretation can be brought down, increasing the data's effective value to the mission. In this context, our aim is to leverage visual data to automatically

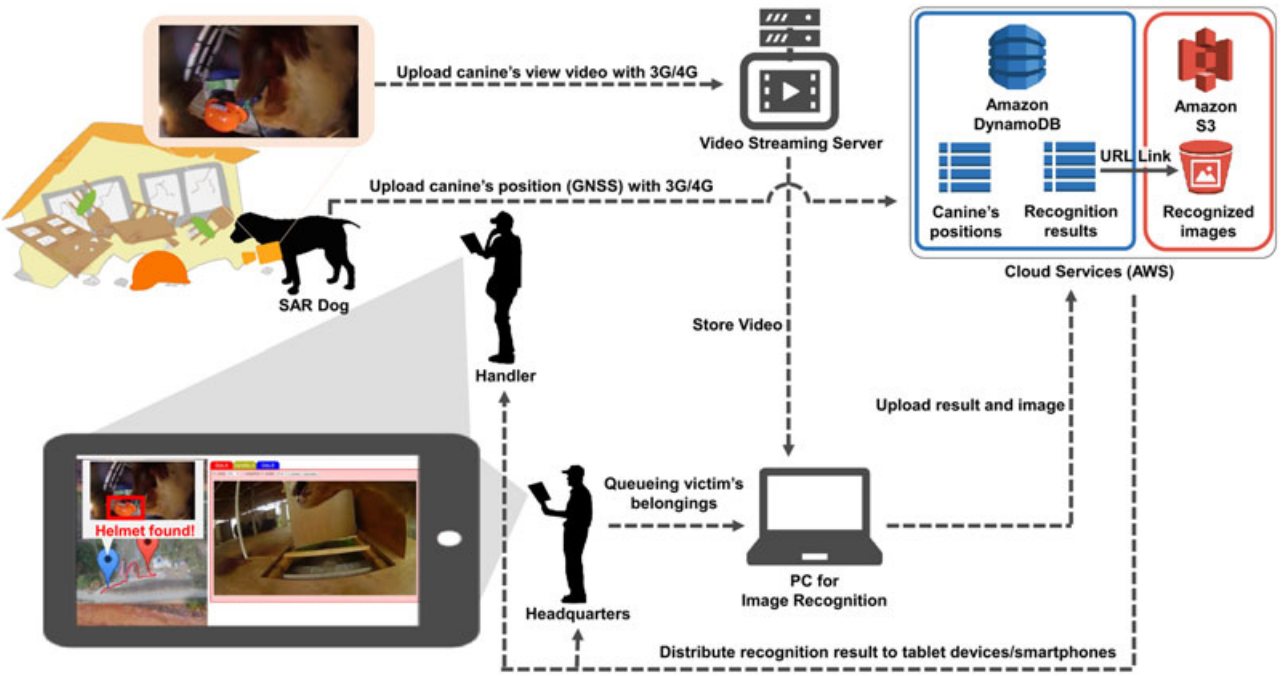


**FIGURE 1** System overview. AWS: Amazon Web Services; 4G mobile connection: fourth generation mobile connection; 3G mobile connection: third generation mobile connection; GNSS: Global Navigation Satellite System; PC: personal computer; SAR: search and rescue; URL: uniform resource locator [Color figure can be viewed at wileyonlinelibrary.com]

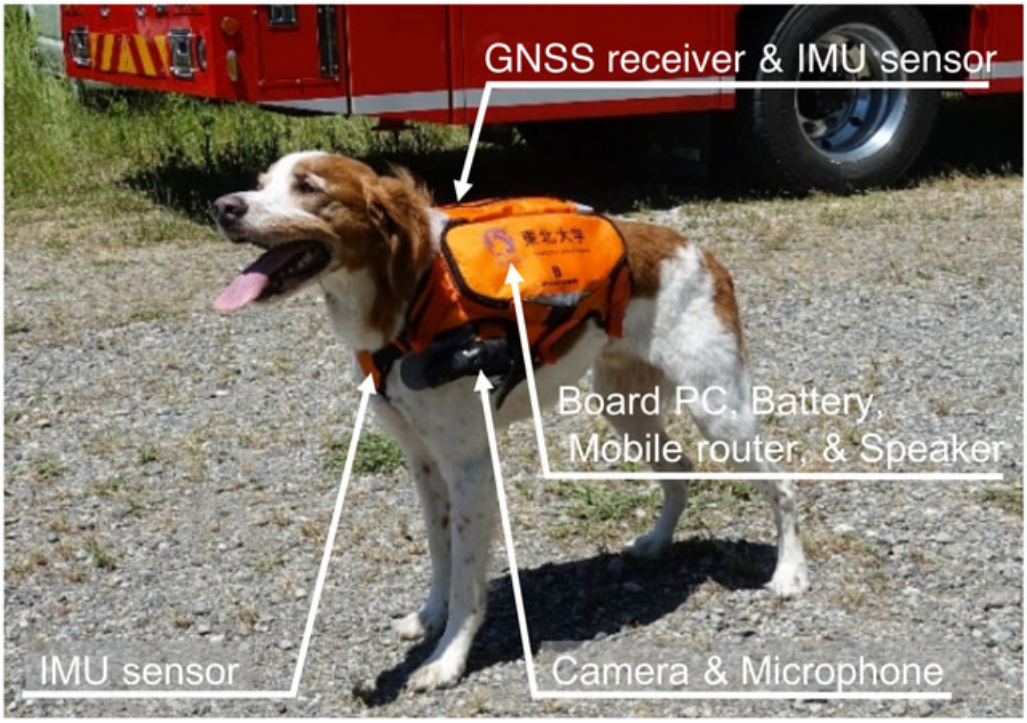


**FIGURE 2** Placement of sensors and other system elements in the cyber-enhanced canine suit. GNSS: Global Navigation Satellite System; IMU: inertial measurement unit; PC: personal computer [Color figure can be viewed at wileyonlinelibrary.com]

identify and locate relevant clues in the field. The "backtrack" recognition functionality of our system, in particular, provides functionality that would be costly to replicate with human cognition alone.

As noted above, rescue dogs' mobility exceeds that of remote-controlled ground robots by a large margin. The fact that rescue dogs can cover ground quickly and move around with high autonomy makes the footage they collect potentially quite valuable for this type of support. This potential has for example previously been leveraged for three-dimensional (3D) scene construction (Tran, Ufkes, Ferworn, & Fiala, 2013).

### 3.1 | Issues in image-based recognition

The problem of automatically detecting objects captured in images has a long history of research in the field of computer vision. However, assuming that images taken by remote-controlled entities are used, challenges unique to disaster response come up. We organise them as follows:

1. *Image quality*: Images captured by an SAR dog might be blurred due to sudden movement or change of direction. In addition, due to bandwidth limitations of the wireless communication network, high image quality cannot always be expected. Countermeasures against these difficulties in image processing are necessary.
2. *Diversity of recognition targets and search environment*: What objects are or are not important varies depending on the mission and the type of disaster environment. Search targets vary on a case-by-case basis, as do search environments. Therefore, it is difficult to collect appropriate training data beforehand. Recognition abilities that are easily adapted to new environments are necessary.
3. *Real-time performance*: For search activities, it is desirable that live footage sent from remote systems can be processed on the spot, in real time. This requires high-speed processing. (Subsequently, finer analysis of the collected footage may be performed offline.)
4. *Integration with information other than images*: It is desirable to present operators with an intuitive overview of when what was found where, integrating different information types. Because image recognition only answers the "what" question, it is necessary to combine it with other modalities to add a "when" and "where" to recognition results.

### 3.2 | Background work in computer vision

A common method for finding prespecified objects in visual scenes is template matching (Brunelli, 2009). Template matching can be made robust against certain types of deformations, but its computational cost increases quickly with the number of deformations considered and the number of targets specified. Also, physical deformability of target objects is hard to account for. We expect deformable targets to be especially common in rescue operations (clothes, bags, headwear, etc.).

Recent years have seen substantial improvements in image recognition. The main drive behind this progress comes from

advances in neural network technology. Convolutional neural networks (CNNs), first proposed in the 1980s and 1990s (Fukushima, 1980; LeCun et al., 1990), have over the past years become the technology of choice for many computer vision applications. CNN-based image recognition showed large improvements compared with traditional methods in one of the leading image recognition competitions ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2012 (Krizhevsky, Sutskever, & Hinton, 2012) and has since dominated much of the field, with all subsequent winners using variations and extensions of the CNN architecture. However, training CNNs for object recognition generally requires large amounts of labelled training data, and long training times, which is an obstacle for our intended use scenario, as we assume recognition targets to be unknown until shortly before operation, or even during operation.

Another strand of research in the field of neural networks is unsupervised learning by means of autoencoder networks. Unsupervised learning has long been studied as a means of finding latent structure in data, which can be applied for a variety of purposes, such as compression, visualisation, and feature extraction (but not, on its own, recognition). Convolutional variants of the autoencoder architecture (Jonathan, Ueli, Dan, & Jürgen, 2011) are particularly apt at compressing and decompressing images, and in the process extract features that can be used for a variety of goals.

### 3.3 | Architecture

Considering the needs and challenges discussed above, we designed an image recognition system with the architecture shown in Figure 3. Designed in consideration of the restriction that labelled data for the task at hand are scarce, this system is designed to allow quick acquisition of target recognition ability from minimal instruction by the operator. It addresses the issues raised above in the following ways:

1. *High diversity of recognition targets and operation environments*: We adopt a semisupervised learning framework. Pretraining provides the ability to identify candidate object regions by means of image segmentation, which facilitates definition of targets by the operator. Once the operator has defined a target, the system uses aggressive supervised learning to quickly learn to recognise it.
2. *Flexibly adjustable recognition ability*: We introduce 'representation nudging' to refine recognition ability where the features acquired by unsupervised training do not suffice to distinguish the target. Representation nudging allows supervised learning to modify feature extraction on the fly, to facilitate specific distinctions.
3. *Real-time operation*: The use of a convolutional autoencoder (CAE) makes it possible to efficiently extract features in parallel on GPU. We also parallelise the recognition pipeline by means of multithreading, so that, for example, the result visualisation for frame 1, feature extraction for frame 2, and preprocessing for frame 3 run simultaneously.
4. *Integration of temporal and spatial information*: By combining the time stamps of video frames with time stamps of recorded GNSS coordinates and spatially mapping recognition results on a map display, the system makes it easy to grasp when what was seen where.

The Core of the system is a CAE. Whereas autoencoders are usually trained unsupervised, we place ours in a semisupervised set-up. Semisupervised learning combines elements of supervised and unsupervised learning. It assumes that the available training data

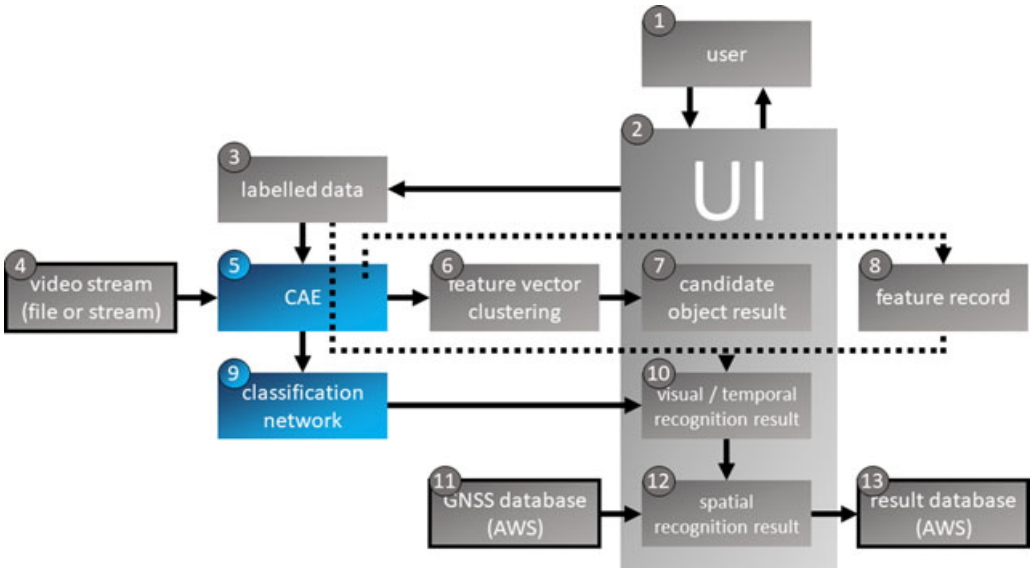


**FIGURE 3** Structure of the recognition system. Arrows indicate information flow. Dotted arrows indicate the pathway of backtrack recognition. Boxes 5 and 9 (blue in the colour version of the article) indicate the neural networks forming the core of the system (detailed further in Figure 4). Boxes with black outline (4, 11, and 13) indicate communication channels with the cyber-enhanced canine suit. Communication via the AWS database is explained in Section 7. AWS: Amazon Web Services; CAE: convolutional autoencoder; GNSS: Global Navigation Satellite System; UI: User Interface [Color figure can be viewed at wileyonlinelibrary.com]

are a mix of labelled and unlabelled data. A common approach is to cluster the data, and then, under the assumption that datapoints in the same cluster belong to the same category, label the unlabelled datapoints using the labels of labelled datapoints falling in the same cluster. Whereas effective, performing clustering with sufficient data to obtain good results becomes computationally expensive. Our approach employs clustering, but in a more limited role. Before we explain the system in detail in the next section, we define some of its core concepts.

1. *Cluster*: Clustering is the partitioning of a set of datapoints on basis of datapoints' similarity. In the present study, each datapoint corresponds to a location on an image, with its data content being a vector of features characterising the image content at that location. Hence, clusters correspond to image regions with similar features.
2. *Category/target:* The object of recognition when we set up the system to recognise a shoe and a hat, 'shoe' and 'hat' are referred to as categories or targets.
3. *Label:* Labels indicating category membership of image regions in the case of labelled data, the label is given by the operator. The purpose of image recognition is to assign accurate labels to unlabelled image regions.

## 3.4 | Dual network architecture

Here, we describe the CAE and classifier network that form the core of the recognition system. A concept image of the dual network set-up and its processing pathways for recognition, supervised training, and unsupervised training is given in Figure 4. This figure corresponds to Boxes 5 and 9 in Figure 3.

### 3.4.1 | CAE

The architecture of the CAE is given in Table 1. The CAE consists of an input layer (IN), three encoder layers (E1 to E3), a bottleneck layer (B), three decoder layers (D3 to D1), and an output layer (OUT). Encoder and decoder layers with the same number have the same spatial resolution. Weights are initialised randomly from a Gaussian distribution with mean 0 and standard deviation 0.04. The hyperbolic tangent activation function is used for all layers.

### 3.4.2 | Classifier network

The classifier network is a small perceptron with two fully connected connection layers. On current GPU hardware, a perceptron of this size can be trained in tens of seconds or less. Matching the dimensionality of individual feature vectors extracted by the CAE, the classifier network's input has size 40 (feature extraction is explained in Section 3.5). The hidden layer has size 24, and the output layer has size equal to the number of targets specified so far. New output neurons, along with their connections, are added dynamically as new targets are added by the user. Weights are initialised randomly from a Gaussian distribution with mean 0 and standard deviation 0.001. As in the CAE, we use the hyperbolic tangent activation function.

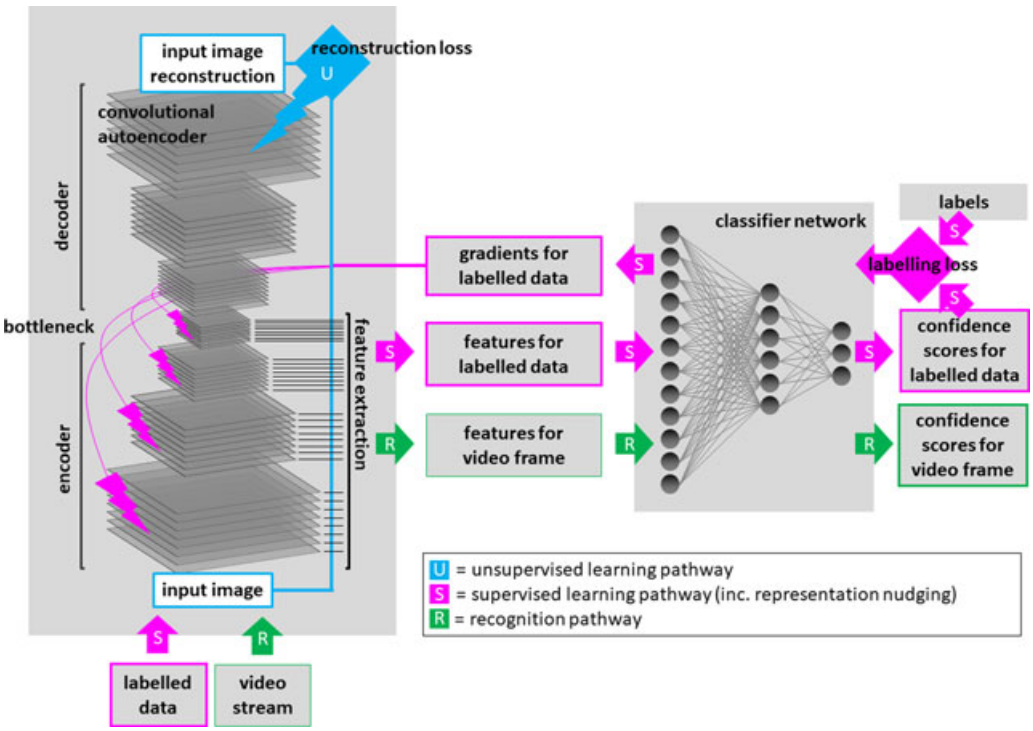


**FIGURE 4** Overview of the dual network set-up, with its training and recognition pathways (networks not to size) [Color figure can be viewed at wileyonlinelibrary.com]

**TABLE 1** Convolutional autoencoder architecture

| Layer | IN | E1 | E2 | E3 | B | D3 | D2 | D1 | OUT |
|---|---|---|---|---|---|---|---|---|---|
| Feature maps | 3 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 3 |
| Convolution | | Down | Down | Down | Down | Up | Up | Up | Up |
| Kernel size | | 5 × 5 | 5 × 5 | 5 × 5 | 5 × 5 | 5 × 5 | 5 × 5 | 5 × 5 | 5 × 5 |
| Stride | | 1 | 2 | 2 | 2 | 2 | 2 | 2 | 1 |

## 3.5 | Extracting feature vectors

Individual video frames typically contain a variety of objects, so recognition should be performed at finer granularity than whole images. We extract feature vectors for a subset of the image's pixels (called *focal pixels* below), arranged in a regular grid. The grid's internode distance $d$ is a system parameter, which should be set in consideration of the image resolution. It was set to 16 in all experiments discussed in this paper. The grid's $x$ and $y$ offsets w.r.t. the image (defined as the $x$ and $y$ distances from pixel 0,0 to the grid's top-left node) are randomised every frame, in the range *(0.5d, 1.5d)*. Combined with temporal smoothing, this offset randomisation helps improve the visual quality of the results.

Since the CAE is fully convolutional, each layer has a 3D spatial structure ($x$, $y$, feature map). For any given focal pixel, we can obtain a feature value from any given feature map in the net as follows: We normalise the pixel's coordinates and the layer's neurons' coordinates into the unit square, find the neuron nearest to the pixel, and assign the neuron's activation value as the pixel's feature value. Gathering the activation values across all feature maps of the layer gives us a feature vector characterising the pixel. Using this principle, we can extract a set of feature vectors for the set of focal pixels from the bottleneck layer. However, restricting ourselves to the bottleneck layer is not necessarily ideal. In our CAE configuration, this would produce feature vectors of length 10 only. Under the assumption of perfect reconstruction, lower layers contain no information that is not also latently represented in the bottleneck layer, but this assumption does not hold in practice. Moreover, the representation in lower layers is different: Less abstract and less compressed. We found it helpful to gather feature values from all encoder layers up to and including the bottleneck layer. We concatenate these into a feature vector of length 40. We hypothesise that feature values from lower layers are helpful in recognising categories defined by low-level texture features. Inclusion of lower layer features also facilitates acquisition and refinement of features as a result of representation nudging, which will be discussed further on.

Each focal pixel's feature vector is determined by a limited region of the input image, the size of which can be computed from the CAE architecture. For our settings, this 'field of view' is 68 × 68 pixels. We refer to such regions as *image patches* below. Note that image patches are over four times larger than the spacing between focal pixels, meaning that the image patches of neighbouring focal pixels overlap substantially.

## 3.6 | Unsupervised training

Our problem setting requires quick, on-the-spot learning. This has implications for the choice of weight updating algorithm. We find that the Manhattan update rule is effective for both quick pretraining and quick category training. The Manhattan update rule is defined as follows:

$$\Delta w_i = \eta * \mathrm{sign}(g_i). \tag{1}$$

Note that the update of weight $w_i$ of connection $i$ is independent of the magnitude of the gradient for $i$, instead being computed from learning rate $\eta$ and the sign of the gradient. Compared with regular stochastic gradient descent (SGD), this has the advantage that learning is fast even when gradients are small. A disadvantage is that training cannot configure network weights at granularity finer than the learning rate, which bounds maximum performance. However, our goal is to obtain adequate performance quickly, rather than maximum performance eventually. Various update rules exist that similarly speed up learning compared with regular SGD, but these generally assume training is towards a fixed goal (whereas we allow targets to change on the fly), availability of some amount of connection update history (which introduces complications when connections are added on the fly), and come with their own instabilities and pitfalls (Wilson, Roelofs, Stern, Srebro, & Recht, 2017). Adapting more refined update rules to the demands of our problem setting is left as an avenue for future study.

We train the CAE module with a learning rate $\eta$ of $5{*}10^{-6}$ on the mean squared error (MSE) over the input and output images.

## 3.7 | Pretraining

The net should preferably be trained in advance on suitable footage. When training a CAE for general use, we find that it is most practical to pretrain on footage that covers the full colour spectrum well and has a good variety of visual elements. We found that general purpose pretraining can be more effective than training on footage overly similar to the operation environment. For example, if the network is pretrained on forest footage and the task environment is a similar forest, performance can still be impaired if the objects targeted for recognition are visually very different from those in the pretraining footage. A net pretrained on purely green and brown forest footage will fail to reconstruct, and hence fail to extract useful features for, a bright blue backpack. The networks used in the evaluation experiments in this paper were pretrained on 8 min of nature footage containing a variety of

**FIGURE 5** Example of source video (left) and its reconstruction by the pretrained CAE (right) [Color figure can be viewed at wileyonlinelibrary.com]

natural environments and colourful animals, left to loop until reconstruction quality was stable and adequate. Visually adequate reconstruction of the input image was obtained in about 15 min of training. As is often the case with CAEs, reconstruction is somewhat grainy and shows some loss of fine detail. There is also a tendency to exaggerate edges. Figure 5 shows a side-by-side comparison of input and output images.

## 3.8 | Feature clustering and category definition

Clustering is used to find regions of similar features, to aid in the defining of targets (Box 6 in Figure 3). We use the Ordering Points to Identify the Clustering Structure (OPTICS) clustering algorithm (Ankerst, Breunig, Kriegel, & Sander, 1999). OPTICS produces hierarchical clusterings without explicit levels. Its output is a pair of lists. One defines an ordering over the datapoints, and the other defines for every pair of neighbouring points in the first list the minimal distance threshold value for which the points fall into the same cluster. We set OPTICS' parameters, $\varepsilon$ and MinPts, to $\infty$ and 2, respectively. Usually, with $\varepsilon$ set $\infty$, computing the list pair would have time complexity of $O(n^2)$. However, we impose a spatial restriction on cluster formation. Because our datapoints are spatially arranged focal pixels, we can impose the restriction that datapoints only connect to datapoints corresponding to adjacent focal pixels. This ensures that clusters are spatially contiguous, and also greatly reduced the number of distance computations necessary for generating the list pair, reducing complexity to $O(n)$. Once computed, we can efficiently compute the clustering that results for any distance threshold. This is particularly useful in target definition, as it makes it possible to let the user adjust the distance threshold as a way of quickly finding a clustering that aligns well with the target object.

With a well-trained CAE producing the feature vectors, the clusters generally latch on to visually similar regions. The result is presented on the GUI (Box 7 in Figure 3). Figure 6 shows a generic example. Most objects consist of one or more visually homogeneous parts, making it possible to specify targets by simply adjusting the distance threshold until a good fit is found, and then assigning a name to the cluster(s) corresponding to the target. We implemented four (closely related) ways of entering labelled data into the system. Each adds labelled data to a local database (Box 3 in Figure 3).

1. *On-the-fly*: At any time, the user can pause the system, select and modify a cluster from the frame they paused on, and assign a label to the cluster. The image patches contained in the cluster are then added to the database as positive examples of the category.
2. *From image files*: Similar as above, except instead of using the current footage, the image data are sourced from an image file.
3. *From stored queries*: Target definitions produced by Methods 1 and 2 can be stored as query files. Loading a query file instantly adds its content to the database. Saving and loading queries should be particularly useful for sharing queries across multiple instances of the system.

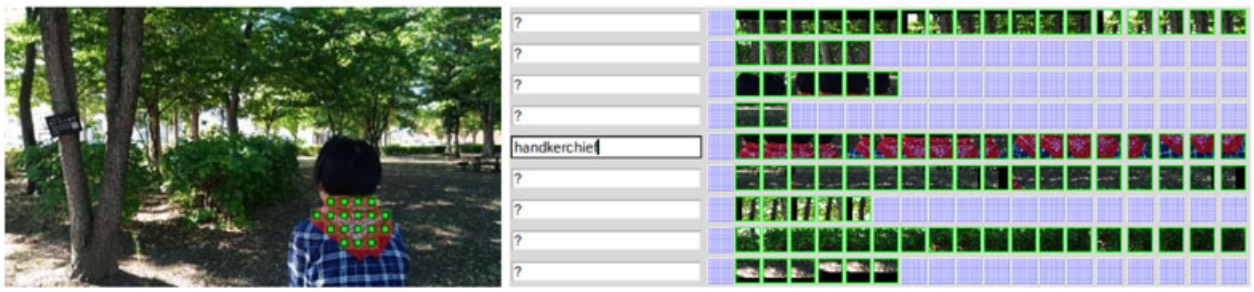


**FIGURE 6** (Left) Example of clustering identifying a candidate object. By adjusting a single slider (controlling the clustering distance threshold), the user can adjust the clustering to align with visually homogeneous image regions. (Right) Cluster list view. The user can assign a label to a cluster by typing the label name in the corresponding text box. Clusters are highlighted by placing the cursor near any of its member focal pixels (marked with green boxes on the video frame), or by placing the cursor over the cluster's row in the cluster list [Color figure can be viewed at wileyonlinelibrary.com]

4. *By answering queries from the system*: For every label in the database, the system keeps a list of image patches that produced the highest recognition scores for that label. This list is continually updated during training. The user can display these lists and indicate which patches indeed belong to the label and which do not. This provides both additional positives (if available) with minimal user effort and also provides highly informative negative examples: As these examples produced high recognition scores for a given category, their nonmembership to that label helps delineate the range of the category in feature space (we elaborate on this point below).

## 4 | RECOGNITION MODALITIES

As noted in the introduction, the system implements two recognition modalities. The practical difference lies in the order of events: The first modality, which we refer to as *forward recognition* below, assumes that targets (i.e., labelled data) are available ahead of system deployment and relies on a short training phase preceding operation. The second modality, which we refer to as *backtrack recognition* below, assumes that a novel target is presented during operation and tries to quickly find occurrences of it in the footage observed up to that point.

### 4.1 | Forward recognition

#### 4.1.1 | Classifier training

As is common, the classifier is trained on fixed-size minibatches of labelled data. Minibatches are randomly selected from the database of labelled image patches. Each minibatch contains 10 positive examples for each category and 10 negative examples for each category for which negative examples are available. Selection is with replacement, so even if there are fewer than 10 (but more than 0) examples in the database, we will train on 10 examples (including duplicates). This avoids diminishing the effect of categories with very few examples. Note that examples here are image patches. A single labelling operation by the user will typically provide tens of examples or more.

Loss for positive examples is the MSE over the labelling output and a one-hot vector indicating the correct category. Loss for negative examples is more complicated. Using the MSE computed over the labelling output and a vector of all zeros (which reduces to the sum of squares over the labelling output) makes intuitive sense but was found to destabilise the classification network. Instead we use the following loss:

$$\text{loss}^- = \max(0,\, p_c - \text{threshold}_c^+)^2, \tag{2}$$

$$\text{threshold}_c^+ = \mu_c^+ - \alpha^* \sigma_c^+, \tag{3}$$

where $c$ is the category the example is a negative example for, $p_c$ the net's confidence score for the example belonging to category $c$, and $\text{threshold}_c^+$ a recognition threshold value for category $c$. This threshold is recomputed every frame from confidence scores for positive examples. There is a feedback loop between threshold adjustment and classifier training. As selective recognition of a given category improves, its threshold increases, which reduces the loss incurred from the category's negative examples. Consequently, the effect negative examples exert on training dissipates as discriminative ability becomes sufficient to distinguish positives and negatives examples (with sufficiency determined by the $\alpha$ parameter). This conditional effect allows us to suppress the response to negative examples while avoiding the destabilising effects of forcing recognition scores for negatives towards zero.

The training loss is the sum of the losses for positive and negative examples. The training rate for the classifier was set to $10^{-4}$ in our experiments.

Training always uses examples from all categories, irrespective of the system's performance on the individual categories. In general, training selectively on cases on which a perceptron performs poorly generally leads to deterioration of performance on ("forgetting" of) the cases excluded from training, due to perceptrons' reliance on distributed representation (French, 1999; Rumelhart, Hinton, & Williams, 1986). This is particularly true with high learning rates and aggressive learning rules, as are used here. Of course it is possible for categories to become obsolete during operation (e.g., due to successful detection of a target of which we assume only a single instance to exist in the field). In this case, the category may be deleted from the system to free up some computational resources.

#### 4.1.2 | Representation nudging

So far we have described the CAE and classifier as separate networks, but gradients from the classifier are propagated back through the CAE down to the image input layer (note that the supervised training pathway in Figure 4 extends to the CAE). In this way, classifier training can modify feature extraction. Whereas features are initially geared towards reconstruction only, gradients from the classifier nudge the CAE towards features that facilitate category discrimination. We refer to this effect as representation nudging. During classifier training, each subnetwork updates weights at its own learning rate.

Unlike the minibatches processed when performing classification, the minibatches used during training do not constitute coherent video frames, but are batches of image patches. Image patches differ in size from video frames, but as the CAE is fully convolutional, it can handle inputs of arbitrary resolution.

#### 4.1.3 | Overfitting countermeasures

Training a regular CNN with very few examples can quickly lead to catastrophic overfitting. Our problem setting assumes that only few examples are available, so the issue of overfitting must be taken into consideration. Here, we discuss features and aspects of the system that counteract overfitting.

Whereas targets are defined as regions on one or a few images, the classifier trains on image patches extracted from

these images. A single act of target labelling act by the user typically provides tens of patches. By collecting examples at this fine granularity, we increase the effective number of examples for training.

The potential for overfitting is also constrained by various aspects of the architecture. The classifier network is small in size, particularly in comparison to the CAE. Using a minimalistic architecture, in addition to allowing for fast training, also limits the potential for overfitting, as it constrains the number of adjustable parameters (i.e., connection weights). Most of the hybrid network's connection weights are located in the CAE. The CAE is trained primarily for image reconstruction, in unsupervised fashion on diverse imagery. This constrains the CAE to produce features that broadly serve reconstruction. The fact that the classifier network operates on such features likely limits its potential for overfitting. Representation nudging allows for the CAE to be tuned for recognition in addition to reconstruction, but is always accompanied by training on the reconstruction loss as well. Furthermore, we expect the use of the Manhattan update rule to limit the potential for overfitting, in a way similar to small network size, as it constrains the precision to which the weights can be tuned.

Finally, the system performs simple data augmentation, as is commonly used in CNN training. We apply data augmentation on each training batch, in the form of rotation and mirroring. Rotation is limited to multiples of 90° to avoid the computational cost and aliasing effects of rotations by arbitrary angles. Augmentation effectively increases the number of examples by a factor eight.

Whereas it is hard to quantify the effect of these factors, the experimental results presented below provide indication that no catastrophic overfitting occurs.

### 4.1.4 | Thresholding

Object recognition operates by assigning class membership probabilities to images or image regions. In practical use, it is often more convenient to have a binary result (membership or nonmembership). When the category set is complete w.r.t. the data (i.e., if every input is guaranteed to belong to one category in the category set), then we can simply assign inputs to whichever category it scored the highest probability for. Common image recognition benchmarks assume such conditions, but in practice we often cannot make this assumption. In the present study, we assume to have only a small number of object categories, and in general most of the visual inputs to the system will not belong to any known category. Hence, we need to set thresholds for recognition. Finding suitable thresholds automatically turned out to be quite difficult. We found that the optimal threshold setting varies between categories, and even per category it varies over the course of classifier training. To handle this volatility without relying on the user to continuously adjust thresholds, we introduced a simple heuristic procedure for setting tentative per-category thresholds, which the user can adjust manually as needed. Figure 7 illustrates how the threshold affects recognition results on a test object.

$$\text{threshold}_c = \max(\text{threshold}_c^+, \text{threshold}_c^-), \tag{4}$$

$$\text{threshold}_c^- = \mu_c^- + \alpha^* \sigma_c^-. \tag{5}$$

Here, $\text{threshold}_c$ is the threshold for category $c$, and $\mu_c^-$ and $\sigma_c^-$ are the mean and standard deviation over all negative examples in the training subset ($\text{threshold}_c^+$ and parameter $\alpha$ were defined in Section 4.1.1). Threshold values are temporally smoothed (default smoothing factor: 0.9) to reduce fluctuations resulting from the random choice of training batches.

This scheme provides a reasonable first guess for threshold values when the difference in appearance between the image used to define target and an occurrence of the target in the footage is small. However, when this difference is large (due to, e.g., very different lighting conditions or viewing angles), confidence scores will often fall below the thresholds computed by this approach. In practice, the most effective way of interpreting the results is to look for peaks in the time-series of per-frame peak confidence scores to find the frames most likely to contain the target, and explore the confidence distribution within each frame to see what area elicited that confidence.

Image regions whose confidence w.r.t. a target exceeds the threshold setting are marked with a box outline and label marker on the GUI (Box 10 in Figure 3, and Figure 7).

We implemented temporal smoothing over confidence values, which helped to obtain cleaner results on test footage and footage from a serpentine robot platform (Fukuda, Konyo, Eijiro, & Tadokoro, 2014). However, in integrating the system with the cyber-enhanced canine suit, we found that SAR dogs' often high pace and wild movement (especially in comparison to slow robotic platforms) limits how much of such temporal smoothing can be applied, as it assumes high similarity between subsequent frames. This limitation is also related to the choice of camera. In the interest of keeping the suit light and compact, the camera used here is a compact model lacking optical image stabilisation. Software image stabilisation was considered, but is computationally expensive, and moreover involves cropping the frames' edges, which may well contain relevant visual information.

## 4.2 | Backtrack recognition

The training scheme discussed so far achieves reasonable performance under challenging restrictions on labelled data availability and training time. However, it still imposes a limitation reducing its practical use: We need to train the system to recognise targets in advance of its deployment. This is generally a given in object recognition systems, but consider the following scenario: A volcanic eruption and ensuing landslide occur in a mountain area popular with tourists. Assuming people may have gotten trapped in the rubble, SAR dogs are deployed to search the area. Initially, no information is available about individuals present in the area at the time of the eruption, but during the course of the rescue mission, information and pictures (e.g., photos individuals in the area sent to friends or

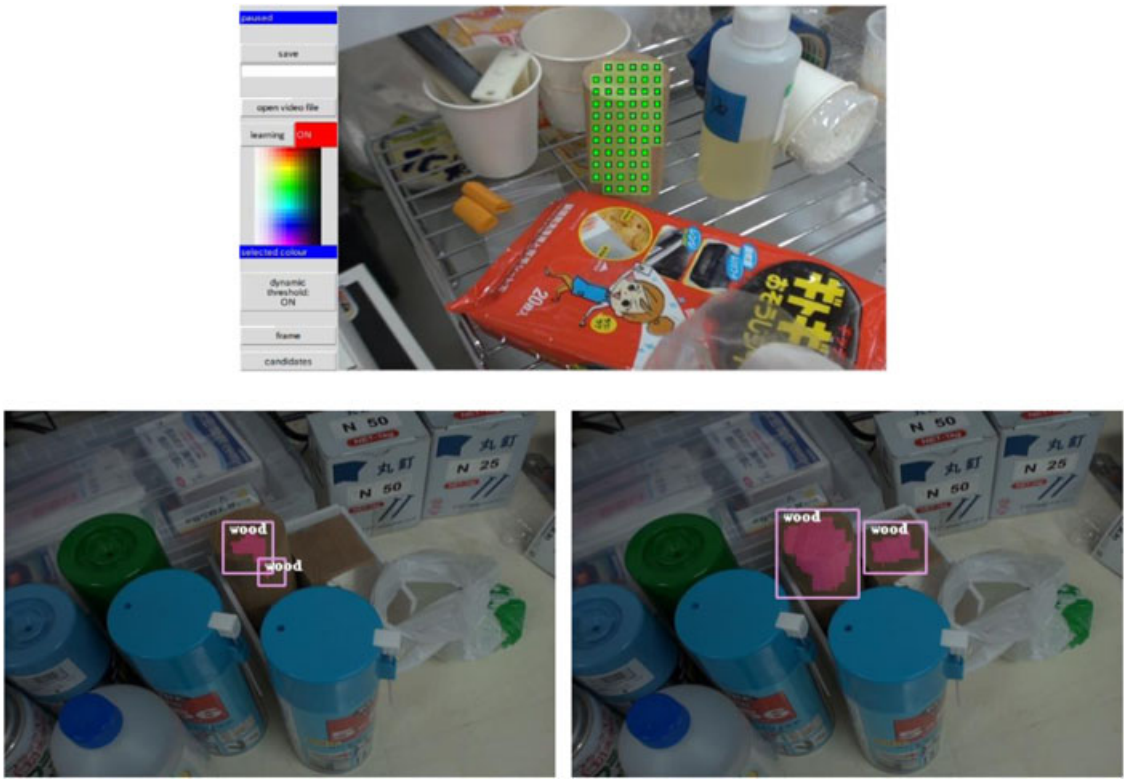


**FIGURE 7** Results obtained for a wooden cylinder test object after placement in a novel context. The target was labelled once (top panel), followed by automatic negative example acquisition from a different scene not containing the target. As the recognition threshold is lowered, we first obtain the result on the left, marking the target. As the threshold is lowered further, we obtain the result on the right. Here, lowering the threshold results in a wooden cube (which presents the same texture on a different shape) being recognised as a second occurrence of the target [Color figure can be viewed at wileyonlinelibrary.com]

family in the hours before the eruption) trickle in, providing visual search targets. In this kind of scenario, it would be very useful to be able to query the system post hoc, to find new targets in the footage seen so far. One could rerun the system on recorded footage, but this would take a fair amount of time. We implemented a backtrack recognition strategy that attempts to find newly defined targets in seen footage in a fraction of the time it would take to process the same footage from scratch. We store the feature vectors extracted from each frame, thereby producing a compressed record (which could be played back as video by processing it through the decoder part of the autoencoder) of the footage seen so far (Box 8 in Figure 3). When a new target is defined, we obtain compressed representations of it through the encoder and compare these to the feature record of the seen footage to find close matches. This comparison in feature space is relatively low-dimensional and extremely well paralellisable. Below we detail the algorithm. The shortcuts represented by this recognition modality are shown as dotted arrows in Figure 3.

Let $Q$ be the set of feature vectors obtained for the image patches defining the query and let $R$ be a database of recorded feature data, along with metadata: $R = \{r | r = (\text{feat}_r, \text{coord}_r, \text{index}_r)\}$, where $\text{feat}_r$ is the feature vector, $\text{coord}_r$ a tuple containing the coordinates of the focal pixel in the frame, and $\text{index}_r$ the index of the frame.

Generally, the number of items in $Q$ will be small ($N_Q < 1{,}000$) and the number of items in $R$ will be large ($N_R \gg 1{,}000$). The distance vector $D$ between the query and all database feature vectors is computed as follows:

$$D(r) = \min\{D'(q, r) | q \in Q\}, \tag{6}$$

$$D'^{(q,r)} = \frac{|q - r|}{\sigma_R}, \tag{7}$$

where $\sigma_R$ is a vector of standard deviations for the feature vector elements, computed over $R$. $D$ gives for each feature vector in $R$ the normalised absolute distance to the nearest element of $Q$. We find the best candidate match using

$$b = \text{argmin}(D). \tag{8}$$

Now, we use $\text{index}_b$ and $\text{coord}_b$ to extract the relevant video frame and mark the focal pixel corresponding to the best match. We also reduce $D$ to a one-dimensional (1D) time-series as follows:

$$T(t) = \min\{D(r) | \text{index}_r = t\}. \tag{9}$$

time-series $T$ is used for evaluation in our evaluation experiments below and is also used in conjunction with GNSS data to plot a spatial

overview of recognition confidence, as will be explained in Section 7.2.

One open issue with backtrack recognition is that it is not compatible with on-the-fly addition of targets for forward recognition if representation nudging is enabled. This is because representation nudging modifies how image patches are mapped to compressed representations. Backtrack recognition works by comparing compressed representations of the search target against compressed representations extracted from seen footage. If the representation format is altered during operation, this comparison becomes incoherent.

## 5 | SYSTEM OPERATION

With the system's architecture and algorithms explained, we now turn to its operation.

The training procedure for forward recognition runs as follows:

1. *Find an image or video frame containing the target*: When defining targets from a video file, pausing the video allows the user to define targets from the paused frame. Targets can also be defined by loading previously saved definitions (in this case, Steps 2 and 3 are skipped).
2. *Find the target in the hierarchical clustering*: This is generally a matter of adjusting the clustering distance threshold parameter (accessed via a slider on the GUI), until a cluster aligns with the target region. Increasing the threshold makes clusters grow and merge, while decreasing it makes them shrink and split. Clusters can be manually refined if necessary by adding and removing individual focal pixels.
3. *Enter a name for the target in the label entry field of the cluster*: For targets comprised of visually distinct parts, Steps 2 and 3 should generally be repeated for each part (assigning the same name to the relevant clusters). Completed target definitions can be saved for future use.
4. *Let the system acquire negative examples*: This can be done in various ways. The simplest is to set negative example gathering to automatic, and let the system run for a while in training mode on footage known not to contain the targets. Alternatively, one can run the system in training mode while occasionally marking candidate image patches (which are automatically collected on basis of their confidence scores) as positive or negative examples. It is also possible (though impractical) to add negative examples manually via the same cluster-and-label process used for positive example definition.
5. Open a video file or stream, and let the system run in recognition mode.

The cluster-and-label process is illustrated in Figure 6. The user can repeat the cluster-and-label process any number of times for different visuals of the target, when available.

The procedure for backtrack recognition is similar, but of course different in order:

1. Run the pretrained system on footage to search.
2. Define a target by means of the cluster-and-label procedure given above, or load a previously stored target definition.
3. Press the backtrack recognition button.

## 6 | QUANTITATIVE EVALUATION

Footage for evaluation of the recognition system was recorded in a rubble field set-up at the Hyogo Prefectural Emergency Management and Training Center, Miki city, Japan (Figure 8). We placed a number of objects people typically wear or carry around in various locations and collected footage over numerous passes through the field.

Whereas the hands-on training style described above is practical for quickly training the system to recognise a given target, 'human-in-the-loop' training poses challenges for quantitative evaluation. To obtain an objective performance measurement, we adopt an evaluation procedure that eliminates user involvement. We pretrain the CAE on unrelated footage (nature footage showing various animals). For every recognition target, we prepare one example image in advance (sourced from footage not used for evaluation). From each image, we make one query using the system running with the pretrained CAE. These are stored and reused for evaluation on five footage files each. The next sections describe the evaluation steps and results per recognition modality

**FIGURE 8** The rubble field at Hyogo Prefectural Emergency Management and Training Center [Color figure can be viewed at wileyonlinelibrary.com]

### 6.1 | Forward recognition

For evaluation of forward recognition, negative example acquisition is automated as follows: We open a file of footage from our data set that does not contain the target, load the relevant queries, and then let the hybrid network train for 350 frames. Starting at Frame 25, and repeating every 25 frames until Frame 300, we let the net automatically accept the candidate examples lined up in the candidate list as negative examples. Once this process is completed, we disable training and evaluate the performance on video files containing the target (Figure 9).

We use temporal annotations. Each footage file's annotation indicates for every frame which targets, if any, are present. This provides a binary time-series for each target. Accuracy is measured as follows: For every target, we track the peak confidence score over each frame (i.e., the highest confidence of the target being present, over all pixels in the frame). We then compute the correlation coefficient over the time-series of confidence scores and the binary time-series derived from the annotation. Higher correlation coefficients indicate stronger correlation between object presence and per-frame peak confidence. As a second evaluation measure, we check whether the peak confidence score over the entire run correctly identifies the target. We do this by finding the pixel assigned the highest confidence across all frames in the footage and visually checking whether this pixel falls on the target. Evaluation results for forward recognition are given in Table 2. For peak hits, Y indicates correct identification of the target and N indicates failure. The peak hit evaluation is motivated by the intended use of the system. When trying to locate a given object on basis of video and GNSS data, the GNSS coordinates of the frame producing the highest confidence can be considered the system's best guess for the approximate object location.

### 6.2 | Backtrack recognition

For backtrack recognition, we run the video file through the pretrained CAE process, without any labelled data in the system. Then, we open the stored query for the target object and run the backward recognition process with it. This produces distance vector $D$ and time-series $T$. Again, we evaluate performance using the correlation coefficient, this time between time-series $T$ and the binary time-series indicating object presence (i.e., the temporal annotation). In forward recognition, higher scores indicate higher confidence, whereas in backtrack recognition, higher scores indicate larger distance and thus lower confidence. To account for this, we use the negative of the correlation coefficient here. For peak hits, we find the minimum value in $D$ and mark the corresponding location in the corresponding frame. We then again visually check whether the marked location falls on the target. Table 3 gives the results.

Backtrack recognition is geared towards speed (processing hundreds of frames per second) and hence simple compared with forward recognition (which does not need to exceed real-time recognition). As is to be expected, this speed comes at a cost. Correlation scores are lower on average than in forward recognition, and we see more peak hit failures. However, despite its simplicity, backtrack recognition still manages to locate the target object in most of our test videos (11/15).

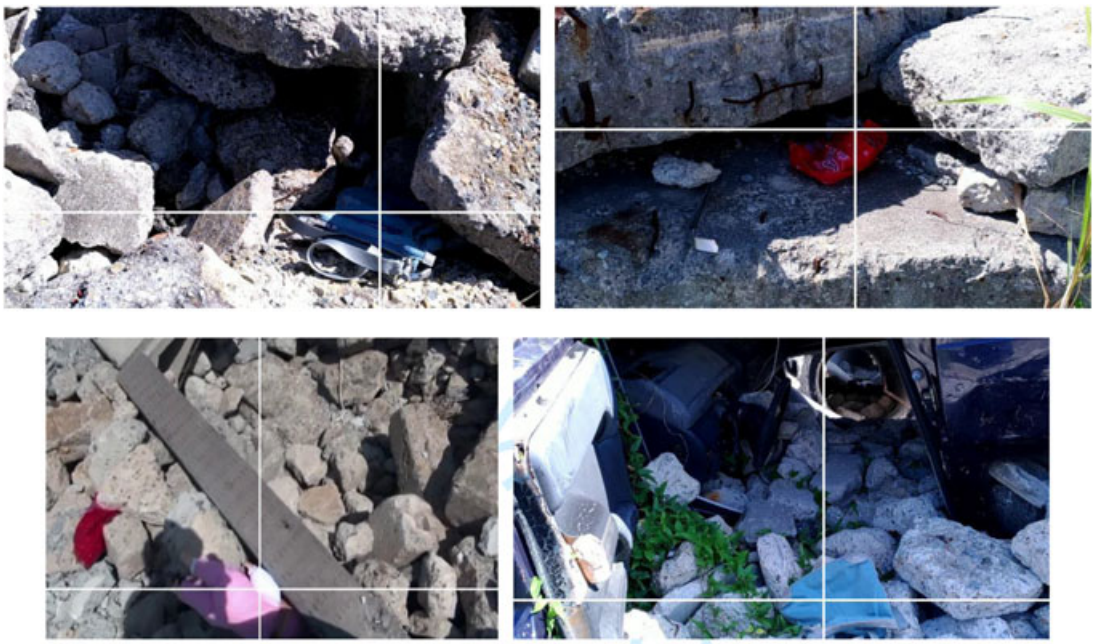

**FIGURE 9** Example results. The crosshairs indicate peak confidence locations over the run (peak hits). The top row shows the results of forward recognition (bag and handkerchief) and the bottom row the backtrack recognition (doll and bag). Resolutions differ due to the use of different cameras [Color figure can be viewed at wileyonlinelibrary.com]

**TABLE 2** Evaluation results for forward recognition

| | Handkerchief | | Shoulder bag | | Doll | |
|---|---|---|---|---|---|---|
| Footage # | Corr. coef. | Peak hit | Corr. coef. | Peak hit | Corr. coef. | Peak hit |
| 1 | 0.60 | Y | −0.23 | N | 0.70 | Y |
| 2 | 0.86 | Y | 0.82 | Y | 0.93 | Y |
| 3 | 0.62 | Y | 0.87 | Y | 0.89 | Y |
| 4 | 0.90 | Y | 0.54 | Y | 0.16 | Y |
| 5 | 0.86 | Y | 0.23 | Y | 0.51 | Y |

Processing times for backtrack recognition are shown in Table 4. These measurements were obtained on a PC with a 3.50 GHz 12 core central processing unit (CPU), an Nvidia Geforce GTX 1080 GPU and 64 GB of random-access memory (RAM). We used a typically sized query of 52 feature vectors for these measurements. In Table 4, 'reading' refers to loading the feature vectors from the record of the frames seen so far. The record is organised in frames, so the time per feature vector is amortised over frames. We processed frames here at a resolution of 640 × 360 with an inter focal pixel distance of 16 pixels and some randomisation of the placement of the top-left focal pixel, producing 811.5 feature vectors per frame on average. 'Distance computation' refers to computation of list *D*. Distances are computed in chunks of 250,000 recorded feature vectors and amortised over this chunk size. 'Postprocessing' refers to the process of building up a list of confidence peaks for display. With settings as given above, this per-vector processing time translates in a framerate of approximately 300 fps. Hence, when recording at 5 fps, backtrack recognition can process a minute worth of seen footage in approximately 1 s.

The shortcomings seen in backtrack recognition illustrate the positive effects of negative data. For example, on Footage 4 for the shoulder bag, the bag sits across the border between a brightly lit area and a dark shadow, giving it an appearance quite different from the image used to define the target. The peak confidence location misses the bag, instead indicating a spot on a chunk of shadowed rubble with colour values close to the image used to define the target. In our evaluation of forward recognition, image patches of such similar content were automatically added to the database during the negative example acquisition step, allowing the system to avoid this misrecognition.

**TABLE 4** Computation times for backtrack recognition

| Processing step | Time cost/feature vector (s) |
|---|---|
| Reading | $1.058*10^{-6}$ |
| Distance computation | $5.029*10^{-7}$ |
| Postprocessing | $2.544*10^{-6}$ |
| Total | $4.105*10^{-6}$ |

## 7 | INTEGRATION WITH CYBER-ENHANCED CANINE SUIT AND FIELD TESTING

We integrated the vision system with the cyber-enhanced canine suit and qualitatively evaluated performance in the field. Integration with the suit requires video transfer from the suit's camera to receiving computers, obtaining GNSS data provided by the suit and combining it with recognition results into a map display, and uploading of recognition results to the result database. We discuss these topics in order, followed by discussion of our experience in the field.

### 7.1 | Video transfer testing

We verified how much delay occurs in video distribution using the mobile phone line. The experimental set-up for evaluation is shown in Figure 10. The video camera was connected to Single Board Computer, and the video was delivered through the mobile phone line. The object to be photographed was a black disk with a rotating arrow and a display. The black disk rotated at 60 rpm and covered about 30% of the image area. By constantly changing a part of the

**TABLE 3** Evaluation results for backtrack recognition

| | Handkerchief | | Shoulder bag | | Doll | |
|---|---|---|---|---|---|---|
| Footage # | Corr. coef. | Peak hit | Corr. coef. | Peak hit | Corr. coef. | Peak hit |
| 1 | 0.763 | Y | −0.401 | N | 0.401 | Y |
| 2 | 0.854 | Y | 0.460 | Y | 0.475 | N |
| 3 | 0.747 | Y | 0.606 | Y | 0.625 | Y |
| 4 | 0.884 | Y | 0.321 | N | 0.202 | N |
| 5 | 0.779 | Y | −0.221 | Y | 0.759 | Y |

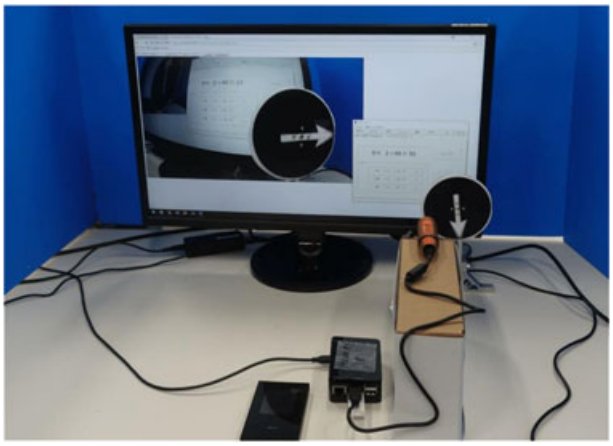

**FIGURE 10** Delay measurement set-up [Color figure can be viewed at wileyonlinelibrary.com]

field of view, compression and transmission load of the video stream (compressed in H.264 format) was obtained. A timer was displayed on the display, and the difference between the current time and the time displayed in the delivery video was calculated.

Distribution delay measurements were performed immediately after the start of video distribution and after 2 min of continuous operation. Values were recorded in a precision of two decimals. We recorded the delay five times for 3G and 4G lines each and calculated the average and standard deviation for each line. The 3G line yielded a 2.7 ± 1.7 s delay at the start of operation and 9.0 ± 6.7 s at the 2-min mark. The 4G line yielded a 0.78 ± 0.07 s delay at the start of operation and 0.81 ± 0.07 s at the 2-min mark. That is, the delay was less than 1 s in 4G lines. In addition, the increase in delay time was 6.3 s on average for 3G lines and 0.02 s on average for 4G lines.

Figure 11 shows examples of video distributed over the 3G line (left panel) and 4G line (right panel). There was no large deterioration in image quality of the black disk in the image. However, on the 3G line, the image occasionally froze.

## 7.2 | Integrating GNSS data and recognition results

For integration between the cyber-enhanced canine suit and recognition system, we configure the recognition system to read

**FIGURE 11** Example frames of video transmitted over 3G line (left) and 4G line (right) [Color figure can be viewed at wileyonlinelibrary.com]

video data received via WebRTC connection on the 4G line, implemented spatial result display functionality, and implemented the communication channel for sending spatially marked recognition results to the suit's main User Interface (UI).

As discussed above, the suit records its GNSS coordinates and uploads this data to the AWS database (Box 11 in Figure 3). The recognition system retrieves this GNSS data and combines it with image recognition results to produce a spatially structured result display. This spatial result display is similar between forward and backtrack recognitions, both of which produce a time-series of per-frame peak confidences. The result display maps these time-series onto a map of the environment obtained via the Google Maps API, locating the frames in space by means of the GNSS data provided by the suit (Box 12 in Figure 3).

Video data and GNSS data are recorded and stored asynchronously. GNSS data are recorded at a frequency of about 1 Hz, so notably lower than video data. To position a frame in space, we take its time stamp (arrival time minus estimated lag), find the entries nearest in time in the GNSS database, and linearly interpolate to obtain an estimate coordinate at which the frame was recorded. The GNSS database is polled at a minimum interval of 0.25 s. We plot the frames as dots on the map, with locations where confidence exceeded a given threshold highlighted. The threshold can be adjusted from the UI, with the map display updating on-the-fly. This result display provides a quick overview of possible locations for the target object. When the user moves the cursor near a dot, the corresponding frame is displayed.

Whenever a backtrack recognition query is processed, we display the full results on the local UI of the recognition PC, and then upload a selection of frames (along with its time-stamp and estimated GNSS coordinates) to AWS for retrieval by and display on the main PC (Box 13 in Figure 3). The selection of frames to upload is made automatically by picking the three frames with the highest peak confidence scores. Frames are marked with an overlay that indicates the region of high confidence in the frame. As discussed above, finding optimal threshold values to determine what is counts as high confidence is difficult. The threshold defaults to the values derived in Section 4.1.4, but can be adjusted on the recognition PC. Note that the threshold setting only affects the extent of the region marked by the overlay; selection of result frames and the determination of the location of peak confidence within those frames do not involve the threshold settings.

One thing to note about the map display is that when a location is assigned high confidence for the target, this does not mean that the target is (likely to be) located *at* that location, but rather that the target is (likely to be) *visible from* that location. It should also be noted that low confidence should not be taken as evidence of absence: Unless the dog turns a full 360° on the spot, we will have only a limited window on the location. Avenues for improvement of these points are discussed in Section 8.

## 7.3 | Field test

We integrated the cyber-enhanced canine suit and recognition system, as shown in Figure 3, and tested it on location in the

**TABLE 5** Specs of recognition laptop PC used in the field test

| | |
|---|---|
| Model | Lenovo P70 |
| CPU | Intel Xeon E3-1505M v5, 2.8 GHz, 8 cores |
| GPU | Nvidia Quadro M5000M |
| RAM | 64 GB |

*Note*. CPU: central processing unit; GPU: graphics processing unit; PC: personal computer; RAM: random-access memory.

ImPACT Tough Robotics Challenge outdoor test field at Tohoku University, Japan. The field spans a grass area, a gravel path, and a rubble area mimicking a collapsed house. The rubble area also contains a Hume pipe covered in rubble. The recognition system ran on a laptop PC with specs as shown in Table 5. This field test focused on backtrack recognition, for the purpose of a demonstration.

Live video footage from the suit of resolution 640 × 480 was received over the WebRTC connection, reduced to 320 × 240 and processed at a rate of approximately 5 fps. Frame skipping was used to keep up with the stream when necessary. A selection of objects to use as targets was brought to the test field, such as handkerchiefs, hats, bags, gloves, and shoes. After confirming the system's basic operation and running a number of test runs with the suit strapped to a life-sized stuffed dog toy, we tested the system using a trained SAR dog. Examples of the spatial result display and successful target recognition are shown in Figures 12 and 13. Examples of misrecognitions are shown in Figure 14.

We confirmed that the recognition system can steadily extract and store feature vectors from the video stream and process backtrack recognition queries in seconds. The ability to define targets on the spot provides good flexibility to respond to various experimental scenarios. After familiarisation with the UI and the clustering slider, defining a target from an image can usually be done in less than a minute.

## 7.4 | Open issues

The WebRTC connection used to stream video from the cyber-enhanced canine suit to the PCs running the UI and image recognition systems dynamically adjusts image quality to maintain a steady framerate. Despite the steady image quality observed in preliminary testing, we found that in the field, large amounts of movement between frames can substantially degrade the image quality, introducing significant compression artefacts. At present, the recognition system does not take such compression variability into account explicitly. Compression damage is hard to avoid without losing acceptable framerates, warranting further investigation into how compression damage affects recognition and whether negative effects can be ameliorated by including variable amounts of (possibly simulated) compression damage in pretraining. We also experienced occasional stream dropouts lasting seconds, possibly related to crowding of WiFi frequencies.

In the field test, we found that performance of the recognition system varies substantially with the light conditions over the course of a winter day. From midafternoon, the camera often catches direct sunlight when facing west, leading to blown-out skies and near-black shadows in the footage. Targets obscured by such shadows are often missed, and the edges of the shadows themselves occasionally elicit spurious responses from the recognition system, making for very challenging conditions. Due to the positioning of the camera and the generally close proximity of the handler, the shadows of the dog and handler are prominent in any footage taken at dusk. Adding a light source of sufficient strength to avoid this would likely add too much weight to the suit, so countermeasures may have to be sought in image preprocessing.

Some of our test objects are primarily characterised by colour, with little other defining features. These objects were found to be particularly likely to produce overrecognition, in the sense that unrelated objects of similar colour were commonly identified as the target. The left pane of Figure 14 gives an example of this problem. Automatic gathering of negative examples in forward recognition was implemented as a countermeasure to this problem, but we have yet to integrate negative examples in backtrack recognition.

The spatial result display shows potential detections of the target, but when inspecting those potential detections one has to

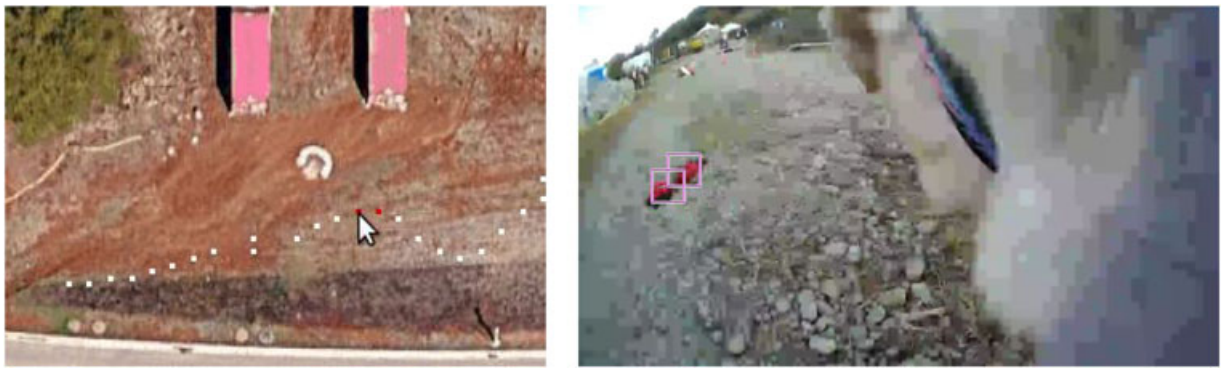

**FIGURE 12** Example result generated with backtrack recognition on a live stream from the suit. The dog's path is plotted in dots on the map in the left window, with the red dots indicating high confidence. Placing the cursor near a dot brings up the corresponding frame, shown here on the right. In this result, the system correctly identified the location from which the target (a pair of gloves) was visible. The SAR dog is visible on the right side of the camera image [Color figure can be viewed at wileyonlinelibrary.com]

**FIGURE 13** Examples of successful detection of a handkerchief (left) and shoe (right) [Color figure can be viewed at wileyonlinelibrary.com]

bring up the corresponding frame by selecting (pointing at) the detection on the map, which was not ideal for quickly determining if and where the target was indeed present among the results. Operation could likely be improved by offering lookup in reverse direction as well (e.g., offering a grid view listing frames of high confidence, with their locations marked on the map as the user selects them).

Finally, we found that the dog sometimes covers more of the video frame than is desirable. An example can be seen in the video frame in Figure 12. It is desirable to have enough of the dog in view to be able to make out what direction the dog's head is facing, but it proves difficult to consistently obtain the right balance between dog visibility and scene visibility. Differences in size and physique between different SAR dogs complicate this issue. Also, it is crucial that the suit be well tolerated and allows the dog to move around freely. This puts limits on how much we can restrict movement of the camera w.r.t. the dog's body. The ergonomics of the suit are a topic of continuing refinement, so we hope to address this issue in a future iteration.

## 8 | CONCLUSION AND FUTURE WORK

We presented an image recognition system for use in disaster scenarios and its integration with the cyber-enhanced canine suit platform. The recognition system facilitates quick learning of targets from limited data and makes providing that data quick and easy. It also provides backtrack recognition functionality, to rapidly find potential targets in seen footage. We evaluated the recognition system on footage gathered in the field, obtaining promising results. Integrated with the suit, the system can automatically plot detections of search targets onto a map display, to provide operators with a quick overview of what was seen where. Challenges remain in recognising objects under harsh light conditions and in footage with heavy compression damage.

At present, backtrack recognition is the more practical recognition modality, but it is also less accurate than forward recognition. In particular, backtrack recognition as presently implemented cannot capture nonlinear relations between categories and feature space, whereas forward recognition does. One direction for future study is to devise an iterative variant of backtrack recognition that allows refinement of the result by means of negative examples, to combine the higher accuracy of forward recognition with the practicality of backward recognition. The present paper considered forward and backward recognition mostly in isolation, but it may be worthwhile to consider interactions between the two. For example, in certain situations (viz., when multiple instances of a target are expected to exist in the field), it could be useful to pass successful detections made by backtrack recognition on to the classification network for training (after confirmation of the detection by an operator).

**FIGURE 14** Examples of incorrect recognition. (Left) Distant cones (blown over by the wind) are incorrectly recognised as the target (red gloves), likely due to similar colour and lack of other defining characteristics. (Right) Operator's shoe incorrectly recognised as the target object (a different shoe). Both have a distinctive white band above a black sole, which may have invited this misrecognition [Color figure can be viewed at wileyonlinelibrary.com]

The current map display shows the dog's trajectory and locations of detections in 2D only. While sufficient for the fields we tested in, elevation can be important in certain disaster sites (e.g., multistoried buildings). GNSS provides elevation measurements, and the positional estimates computed from the IMU do include elevation. However, at present, this elevation information is not visualised beyond simple numerical display of the estimated elevation. How to present this information intuitively without complicating or weighing down the GUI remains as an open issue.

In this study, we have not addressed the problem that disaster sites may be deprived of wireless communications. The suit depends on the availability of a wireless network to communicate its sensor readings to the main UI and recognition system. The signal range of a simple ad hoc network will often be insufficient. Provision of wireless infrastructure at disaster sites is a field of active research in disaster robotics. It has been demonstrated that drones can be used to set up multihop networks that allow for remote communication with out-of-sight robots in otherwise communication-deprived locations (Kagawa et al., 2017). Similar solutions should be viable for the cyber-enhanced canine suit platform, although this issue remains to be explored.

We have not discussed the issue of operating with multiple dogs in the field simultaneously, but such functionality is supported, and a topic on ongoing further development. At present, the suit's main UI supports operation with up to three suits simultaneously. Recognition for multiple suits is possible by running multiple instances of the recognition system (specifying a suit ID to determine which stream and GNSS data to obtain). As recognition is computationally intensive, these instances could be run on separate laptops, or in parallel on a sufficiently powerful workstation located off-site or in a mission vehicle. Functionality for handling multiple suits with a single instance of the recognition system (splitting the framerate between suits) is being considered. Furthermore, the present implementation supports saving and loading of search targets, but scenarios where multiple instances run in parallel call for functionality to automatically share targets between instances.

With regard to result display, we noted that in the current implementation, recognition results plotted on the map indicate locations from which the target is visible, rather than the actual object location. Integration of orientation data and object distance estimates could be used to mark approximate object locations instead, which would be more intuitive. Relatedly, we are considering the use of a footage from a 360° camera present in some recent iterations of the suit. One motivation for the present system is that it may help spot objects overlooked by the easily distracted human visual system. In the present paper, we used footage from a front-facing camera, meaning that the camera generally sees a similar scene as the dog, and, to a lesser extent, the handler. Use of 360° vision would allow our system to process view angles that the dog and handler never see, thereby greatly increasing its potential.

## ACKNOWLEDGEMENTS

We thank four anonymous reviewers for their insightful comments. This study was partly funded by ImPACT Program of the Council for Science, Technology and Innovation (Cabinet Office, Government of Japan).

## ORCID

*Solvi Arnold* http://orcid.org/0000-0003-4342-9344
*Kazunori Ohno* http://orcid.org/0000-0003-3958-2901
*Ryunosuke Hamada* http://orcid.org/0000-0002-7506-0230
*Kimitoshi Yamazaki* http://orcid.org/0000-0002-4096-3288

## REFERENCES

Ankerst, M., Breunig, M. M., Kriegel, H.-P., & Sander, J. (1999). OPTICS: Ordering points to identify the clustering structure. In Delis, A., Faloutsos, C., Ghandeharizadeh, S. (Eds.), *ACM SIGMOD International Conference on Management of Data*, 49–60. New York: ACM Press.

Bozkurt, A., Roberts, D. L., Sherman, B. L., Brugarolas, R., Mealin, S., Majikes, J., … Loftin, R. (2014). Toward cyber-enhanced working dogs for search and rescue. *IEEE Intelligent Systems*, *29*(6), 32–39.

Brunelli, R. (2009). *Template matching techniques in computer vision: Theory and practice*. Hoboken, NJ, USA: Wiley

Ferworn, A. (2009). Canine augmentation technology for urban search and rescue. In Helton, W.S. (Ed.), *Canine ergonomics—The science of working dogs*. Boca Raton, Florida, USA: CRC Press Taylor and Francis Group.

Ferworn, A., Sadeghian, A., Barnum, K., Rahnama, H., Pham, H., Erickson, C., … Dell'Agnese, L. (2006). Urban search and rescue with canine augmentation technology. *IEEE/SMC International Conference on System of Systems Engineering*, 334–338, Los Angeles, CA, USA: IEEE Press.

Ferworn, A., Waismark, B., & Scanlan, M. (2015). CAT 360—canine augmented technology 360-degree video system. *2015 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*. West Lafayette, Indiana, USA: IEEE Press. https://doi.org/10.1109/SSRR.2015.7443003

Ferworn, A., Wright, C., Tran, J., Li, C., & Choset, H. (2012). Dog and snake marsupial cooperation for urban search and rescue deployment. *2012 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, 1–5. College Station, Texas: IEEE Press. https://doi.org/10.1109/SSRR.2012.6523887

French, R. M. (1999). Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*, *3*(4), 128–135. https://doi.org/10.1016/S1364-6613(99)01294-2

Fukuda, J., Konyo, M., Eijiro, T., & Tadokoro, S. (2014). Remote vertical exploration by active scope camera into collapsed buildings. *Proceedings of the 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Chicago, IL, 1882–1888. Piscataway, NJ, USA: IEEE Press. https://doi.org/10.1109/IROS.2014.6942810

Fukushima, K. (1980). Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, *36*(4), 193–202.

Kagawa, T., Ono, F., Shan, L., Takizawa, K., Miura, R., Li, H.-B., … Kato, S. (2017). A study on latency-guaranteed multi-hop wireless communication system for control of robots and drones. The 20th *International Symposium on Wireless Personal Multimedia Communications (WPMC)*. Bali, Indonesia. https://doi.org/10.1109/WPMC.2017.8301849

Kalouche, S., Rollinson, D., & Choset, H. (2015). Modularity for maximum mobility and manipulation: Control of a reconfigurable legged robot

with series-elastic actuators. *2015 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, College Station, Texas, USA: IEEE Press.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems, 25*, 1097–1105.

LeCun, Y., Boser, B., Denker, J. S., Howard, R. E., Habbard, W., Jackel, L. D., & Henderson, D. (1990). Handwritten digit recognition with a back-propagation network. *Advances in Neural Information Processing Systems 2*, 396–404.

Lee, W., Kang, S., Kim, M., & Park, M. (2004). Robhaz-dt3: Teleoperated mobile platform with passively adaptive double-track for hazardous environment applications. *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 33-38, Sendai, Japan: IEEE Press.

Jonathan, M., Ueli, M., Dan, C., & Jürgen, S. (2011). Stacked convolutional auto-encoders for hierarchical feature extraction. In Honkela, T., Duch, W., Girolami, M., Kaski, S. (Eds.), *Artificial Neural Networks and Machine Learning—ICANN 2011, Lecture Notes in Computer Science, 6791*, Espoo, Finland, 52–59, Berlin, Heidelberg: Springer.

Murphy, R., Kravitz, J., Stover, S., & Shoureshi, R. (2009). Mobile robots in mine rescue and recovery. *IEEE Robotics & Automation Magazine*, *16*(2), 91–103.

Nagatani, K., Kiribayashi, S., Okada, Y., Otake, K., Yoshida, K., Tadokoro, S., … Kawatsuma, S. (2013). Emergency response to the nuclear accident at the Fukushima Daiichi Nuclear Power Plants using mobile rescue robots. *Journal of Field Robotics*, *30*(1), 44–63.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning internal representations by error propagation. In D. E. Rumelhart, & J. L. McClelland (Eds.), *Parallel distributed processing: Explorations in the microstructure of cognition* (1, pp. 318–362). Cambridge, Massachusetts: MIT Press.

Sakaguchi, N., Ohno, K., Takeuchi, E., & Tadokoro, S. (2013). Precise velocity estimation for dog using its gait. *Field and Service Robotics, 105*, 515–528.

Tran, J., Ferworn, A., Gerdzhev, M., & Ostrom, D. (2010). Canine assisted robot deployment for urban search and rescue. *IEEE Safety Security and Rescue Robotics*, Bremen, Germany, 1–6. Piscataway, NJ, USA: IEEE Press. https://doi.org/10.1109/SSRR.2010.5981564

Tran, J., Ufkes, A., Ferworn, A., & Fiala, M. (2013). 3D disaster scene reconstruction using a canine-mounted RGB-D sensor. *2013 International Conference on Computer and Robot Vision*, Regina, Saskatchewan, Canada, 23–28. Piscataway, NJ, USA: IEEE Press. https://doi.org/10.1109/CRV.2013.15

Wilson, A. C., Roelofs, R., Stern, M., Srebro, N., & Recht, B. (2017). The marginal value of adaptive gradient methods in machine learning. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (Eds.), *31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA. Advances in Neural Information Processing Systems 30*, 4148–4158. Curran Associates, Inc.

**How to cite this article:** Arnold S, Ohno K, Hamada R, Yamazaki K. An image recognition system aimed at search activities using cyber search and rescue dogs. *J Field Robotics*. 2019;36:677–695. https://doi.org/10.1002/rob.21848