An Object Recognition System for Disaster Robotics - UI Design and Platform Integration -

○Solvi Arnold (Shinshu University) Kimitoshi Yamazaki (Shinshu University)

1. Introduction

Disaster response work is often dangerous, resources are limited, and time is of the essence. Recent years have seen increasing attention for robotics' potential for disaster response support [1]. Within this context, computer vision plays an important supportive role. The vast majority of disaster response robot platforms house at least one camera. Video footage from the robot is an important source of information about the operation area. Consider a setting where numerous units of various robot platforms are deployed in a disaster area. A substantial amount of video data accumulates as the robots traverse the site. Response teams can have visual targets relevant to their operations. One can for example think of hats, bags, and other personal items known to have been carried by missing persons believed to have been at the site. It can be challenging for operators to attend to all visual information coming in. This is especially true for platforms with multiple cameras or 360 degree cameras, but even with a single view, human vision is easily distracted in complex scenes, and operators' attention will generally be focused on robot control. If we further consider that there can be a multitude of targets, one should expect to run into the limits of operators' attentional resources. Targets may also become known partway into a mission. In this case, it would be valuable to be able to trace back quickly through the gathered footage for potential matches. Considering the above, search operations may benefit from an automatic recognition system operating on the robots' video streams. The present paper presents two advances in our development of such a system [2]. The first is improved recognition logic that incorporates context scores (Section 2). The second is development of a user-friendly UI for quickly and intuitively defining recognition targets (Section 3). We also discuss integration with two robot platforms and evaluation experiments in the field.

2. The recognition system

Here we outline the recognition system, and its integration with two robotic platforms. A recognition system for disaster response should take into account the sudden and volatile nature of response operations. Recent years have seen great progress in object recognition capabilities with the rise of convolutional neural networks, but neural networks typically require extensive training on large amounts of data. In a disaster setting, we cannot expect to know the area or recognition targets in advance. We cannot expect to have much data or even time for training the system. To minimize deployment preparation time, our system employs unsupervised pre-training on generic video footage to obtain a generic feature extractor, and performs recognition by comparing features extracted from the video feed against features of the search targets. This strategy makes it possible to add new search targets at any time, without additional training.

Our feature extractor is a fully convolutional autoencoder (fCAE). Using a fully convolutional architecture (instead of the more common architecture with fully connected layers in the middle) has two important merits for our purpose. The first is that fully convolutional architectures can handle variable input resolutions. This is useful in a setting where we have to account for various robot platforms with various camera configurations. The second is that convolutional layers retain spatial structure, meaning that for any given pixel in the input we can find, in any layer of the network, a set of neurons that codes specifically for that pixel and its surroundings (however, for feature maps of smaller resolutions than the input image, the correspondence from pixels to neurons is many-to-one).

From the activation pattern that results from passing the input image through the encoder part of the autoencoder once, we can extract feature vectors for any number of pixels. We extract activation values from all hidden layers up to and including the bottleneck layer, to obtain features ranging from low-level visual information at fine spatial granularity to high-level visual information at rough spatial granularity. For more details on the feature extraction method, we refer to [2]. The CAE is pretrained on nature footage unrelated to the task environment (we found that nature footage provides a rich variety of colours, shapes and, textures, making it effective for general-purpose pre-training).

For each input (video frame or still image) we extract feature vectors for regularly spaced points on the image, obtaining a grid of feature vectors. Feature extraction is implemented as part of the same computation graph as the fCAE, and performed fully on GPU. The feature vectors are used for recognition, but also for target definition (we return to this point in Section 3).

When the user defines a recognition target, this definition internally consists of a set of feature vectors. To obtain a concise target representation we perform clustering on this set, using the OPTICS hierarchical clustering algorithm [3]. We find the first slice of this hierarchy that marks half or more of the feature vectors as belonging to a cluster

(the remaining vectors are considered noise). For each cluster c in this slice, we compute its mean vector c_{mean} and a range vector c_{range}. The range vector consists of the distances between the minimum and maximum values for each vector element, divided by 2, over all vectors in the cluster. The set of cluster means and ranges provides a concise target representation. It is stored along with a user-provided label as a *query*.

To determine the likeliness of a query's target being present in a given video frame, we compute match scores for the feature vectors from the frame with respect to the query. The match score combines two elements: a local proximity score measuring each feature vector's distance to its nearest cluster (accounting to cluster ranges), and a frame-level context score measuring the extent to which the query's clusters are present in the frame. The local proximity score between a cluster c of the query and a feature vector f extracted from the frame is computed as follows:

$$d_{c,f} = mean\left(max\left(0, |f - c_{mean}| - c_{range}\right)\right) \quad (1)$$

The context score for a frame producing feature vector set F w.r.t. a query characterised by cluster set C is computed as follows:

$$ctx_{C,F} = \frac{1}{|C|}\sum_{c \in C} min\{d_{c,f} \mid f \in F\} \quad (2)$$

The match score for a feature vector $f \in F$ w.r.t. a query characterised by C is then given by:

$$\text{match}_{f,C} = 1 - ctx_{C,F} - min\{d_{c,f} \mid c \in C\} \quad (3)$$

The context score makes it possible for locations across the image to boost one another's match score. We show an example of this effect in Section 4. Match score computation is highly paralellisable, so this process, too, is run on the GPU. Given match scores, we can obtain binary recognition results by simple thresholding. However, in practice relative match scores themselves are of more practical use, as we will discuss below. In the discussion below, we will refer to the maximum over the match scores for all feature vectors from a given frame as that frame's match score.

3. User Interface and operation

When operating in conjunction with a live robot platform, the main function of the recognition system's UI is query definition. Recognition results are sent to the main UI of the platform and displayed there in integrated fashion alongside other information from other sensory data. Integration with two platforms is discussed in Section 4.

Targets can be defined from various sources: still images, live footage, and video files. In practical use, we expect still images to be the most common source. As an example, given the ubiquity of cameras and communication services, friends or relatives may be able to supply a snapshot of a missing individual from shortly before disaster struck. In case video or multiple images are available, more robust queries may be construed by including multiple angles in the target definition.

Figure 1 shows the UI. When the system is run, the main window shows the video feed (be it live or a file). To define a query from an image, the user clicks the "load image" button and selects the file via a standard file opening dialog. The main window then shows the image. When working from video, the feed can be paused at any time to obtain a frame, which is then treated as a still image.

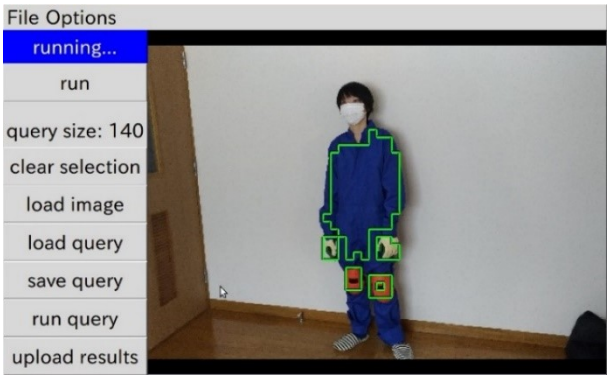

Figure 1. Snapshot of the target definition UI. This snapshot shows a compound selection defining a worker outfit consisting of a jumpsuit, gloves, and knee-pads.

By passing the image through the fCAE, we obtain a grid of feature vectors as explained above. We divide the image into small square regions (*patches* below), each loosely corresponding to one feature vector. We say loosely because each vector incorporates information from a view port substantially broader than the size of the patch, and is focused specifically on the pixel at the centre of the patch. Patches just provide for intuitive visualisation.

The user places the mouse cursor on an image region belonging to the target, and then scrolls up using the mouse wheel, or presses the up arrow on the keyboard, to create a selection. Scrolling (pressing) up grows the selection area, and scrolling (pressing) down shrinks the selection area. The unit of selection here is one patch. The initial selection consists of just the patch at the cursor's location, which we will call the *root patch* below. To determine the next patch to add when growing a selection, we find the set of non-selected patches adjacent to a selected patch, and select the one with the smallest distance in feature space to the feature vector of the root patch. Conversely, to determine the next patch to remove when shrinking the selection, we find from the set of selected patches the one with the largest feature distance to the root patch. We restrict selections to be contiguous areas, so removing one patch may have the effect of removing additional patches.

This selection strategy makes it easy to quickly select regions of similar colour and texture, as selections typically grow to fill out visually homogenous areas before

extending to dissimilar areas.

The system allows making multiple selections by simply moving the cursor to a new location and growing a new selection there. Selections can be discarded by right clicking in the root patch (or shrinking the selection to size 0). The user can also discard all selections at once using the "clear selection" button. In figure 1 we see multiple selections applied to define a compound target.

Once the relevant selections have been made, the user presses the "run query" button to generate a query from the selection and run it (discussed below), or the "save query" button to generate the query and store it as a file. In the latter case, the user is also asked whether to also run the query after saving it. Aside from query reuse at a later time, saving queries is particularly useful for sharing targets between different instances of the system. Infrastructure to further facilitate such sharing may be introduced at a later date. Query files are loaded using the "load query" button.

When a query is run, the user is presented with a prompt to select the recognition mode. Choices here are forward recognition, backtrack recognition, or both. The next section discusses these modes.

For the example in Figure 1, the entire target definition process (from file selection to recognition mode selection) took about 40 seconds in our field experiment.

4.1 Forward recognition

Forward recognition here refers to the common style of recognition where a system runs its recognition processes on each incoming frame in a video feed. In our system, this amounts to applying feature extraction on the frame, followed by computation of the matching scores with respect to each active query. Results are then optionally sent to the robot platform's main UI (see Section 4.3).

4.2 Backward recognition

Backward recognition refers to recognition 'back in time', i.e. recognition of targets in video frames seen before the query was run. To make this possible, the system records the feature vectors extracted from each frame, storing them to the hard-disk. When a query is run in backtrack mode, the stored feature vectors are loaded and match scores are computed for batches of frames. The score calculation is very lightweight compared to feature extraction, and can be performed for many frames in parallel on the GPU, making this backtrack search highly efficient. For a typical system configuration and query, computing the match scores took 7.5×10^{-4} seconds per frame on average, amortised, on an NVIDIA QuadroP3000 GPU. A selection of results is then sent to the main UI (see Section 4.3).

4.3 Result selection

Suitable conditions for sending results depend on the robot platform (e.g. how the results are integrated in the main UI, available bandwidth). In the present implementation we find local peaks in the time-series of the frame match scores, that exceed a minimum threshold and are at least 3 seconds apart (measured by the frames' time-stamps in the case of backtrack recognition), and are separated from the preceding and subsequent peaks by sufficiently deep score dips. This logic is designed to limit the number of results sent and avoid sending near-duplicate results (as often occur on subsequent frames, in particular with slow-moving robots). This in turn is motivated by the result display style of the platforms we have so far integrated our system with. Rather than streaming recognition results alongside the camera view, possible detections are displayed as markers in a spatial view that integrates data from various perception modalities. The corresponding result frame can be brought up by clicking on the marker. A list of result frames, sorted by match score, can also be brought up. Given this display style, stretches of near-duplicate results for subsequent frames would just clutter the display and slow down the user's comprehension of the presented data. The selection logic is further motivated by the practical meaning of the results themselves. Generally, scores will be higher for frames showing the target than for frames not showing the target under otherwise similar conditions, but varying conditions do produce varying scores. For example, if the target appears in light conditions substantially different from the target definition, we can expect low match scores. However, we should still see a bump in the score's time-series that can identify the target. Hence we are more interested in relative scores (bumps) than in the absolute values. Selecting results on basis of local peaks, and then sorting them by score provides a useful presentation of potential detections. False positives may be sent (as is desirable; the harm potential of false positives is far less than that of false negatives), but results with higher match scores "bubble up" to the top of the result display as they occur. The top result for a given target should be interpreted as the frame the system deems most likely to contain the target across the frames processed so far.

For the selected results, we determine bounding boxes as follows. We mark each image patch whose corresponding match score exceeds 0.99 times the frame's match score, and then find the bounding boxes that encompass each connected region of marked patches. This threshold setting was found to be rather conservative, generally producing bounding boxes smaller than the targets.

4. Integration and experiments

In this section we discuss integration and field experiments with two robotic disaster response platforms: the Cyber Enhanced Canine platform [4] [5] and the Active Scope Camera platform [6]. Field experiments were performed at the June 2018 ImPACT field evaluation.

5.1 Cyber Enhanced Canine

The Cyber Enhanced Canine platform (CEC) [4] [5] is a suit designed to be worn by a rescue dog, equipped with numerous sensors: camera, microphone, GNSS, and

sensors to quantify the dog's physiological state. The platform's main UI is a browser application designed to be used on a tablet or smartphone. Video footage is streamed live from the suit over a WebRTC connection to the main UI and the recognition system. Timestamped GNSS coordinates are uploaded to a database on an AWS server. The recognition system retrieves GNSS coordinates and performs simple coordinate interpolation in order to locate video frames in space (frames are received at much higher frequency than coordinates). Results of the recognition process (metadata and video frames) are uploaded to the AWS server, from which they are retrieved for display on the main UI. Metadata includes interpolated GNSS coordinates, match scores, and bounding boxes, among others.

The field experiment with the CEC tested backtrack selection on three items, laid out along a course where the CEC was to discover two victims (participants hidden in shelters). As the dog ran the course, the camera picked up visuals of the target items. For each item, backtrack recognition correctly produced a frame marking the target as its top-ranked result. Results are shown in Figure 2. In this experiment, lighting conditions (sunny) and visual backdrops (only limited distractors) were favourable. However, the recognition system had to contend with substantial compression damage introduced by the live-stream. This produced visual discrepancy, as compression damage was not present in the images used to define the targets.

Figure 2. Successful backtrack recognition of three targets (gloves, hat, shoes) in live footage from the CEC.

5.2 Active Scope Camera

The Active Scope Camera (ASC) [6] is a snake-like robot designed to search for survivors in collapsed structures. It is equipped with an array of sensors including a camera, microphone, and tactile sensors. The ASC is a wired system (an important aspect of its mobility is its ability to hover by means of a downward jet of air, which requires physical connection to a source of pressurised air). Inter-system communication (video reception and results publishing) are implemented in ROS, and occur over a local wired connection.

The field experiment with the ASC simulated search for victims in a collapsed plant, using forward recognition. We defined the target on basis of an image showing the standard outfit of the workers in the plant (Figure 1). The ASC navigated a course through the simulated collapsed structure, to a space where a set of worker's clothes was placed. The lighting conditions in this experiment turned out very challenging. At the time of the experiment, bright sunlight fell into part of the space. This produced a combination of overexposed and underexposed areas. Whereas each individual element of the outfit presented a notably different appearance under these conditions than in the image used for target definition, their combination still elicited the highest match scores over the course of the experiment, so that the top-ranked result by the end of the experiment correctly identified the target (Figure 4).

Figure 3. Successful detection of a worker's outfit in live footage from the ASC, in a simulated disaster setting under challenging light conditions.

5. Conclusions

We presented recent developments in our object recognition system for use in disaster response: context-enhanced recognition logic, a user-friendly UI for target definition, and integration with robotic platforms. In future work, we aim to integrate the target definition UI as a module into the main UIs of the robot platforms.

Acknowledgements

This work was partly funded by ImPACT Program of the Council for Science, Technology and Innovation (Cabinet Office, Government of Japan)

References

[1] 革新的研究開発推進プログラム ImPACT - タフ・ロボティクス・チャレンジ. http://www.jst.go.jp/impact/program/07.html

[2] Solvi Arnold, 公俊山崎: "畳み込み自己符号化器を用いた対話的学習に基づく災害対応のための画像認識システム," *第 22 回ロボティクスシンポジア予稿集*, pp. 211 – 212, 2017.

[3] Mihael Ankerst, Markus M Breunig, Hans-Peter Kriegel, Jörg Sander: "OPTICS: Ordering Points To Identify the Clustering Structure," *ACM SIGMOD international conference on Management of data*, pp. 49-60, 1999.

[4] 和則大野: "サイバー救助犬," *日本ロボット学会誌*, 35(2): 97-100, 2017.

[5] 和則大野: "サイバー救助犬のためのタフセンシング技術," *日本ロボット学会誌*, 35(10): 716-719, 2017.

[6] Junichi Fukuda, Masashi Konyo, Takeuchi Eijiro, Satoshi Tadokoro: "Remote Vertical Exploration by Active Scope Camera into Collapsed Buildings," *Proceedings of the 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1882-1888, 2014.