# Integrating Automated Object Detection into Mapping in USARSim

Helen Flynn, Julian de Hoog and Stephen Cameron
Oxford University Computing Laboratory
July 2009

*Abstract*— **Object recognition is a well studied field of computer vision, and has been applied with success to a variety of robotics applications. However, little research has been done towards applying pattern recognition techniques to robotic search and rescue. This paper describes the development of an object recognition system for robotic search and rescue within the USARSim simulator, based on the algorithm of Viola and Jones. After an introduction to the specifics of the object recognition method used, we give a general overview of how we integrate our real-time object recognition into our controller software. Work so far has focused on victims' heads (frontal and profile views) as well as common objects such as chairs and plants. We compare the results of our detection system with those of USARSim's existing simulated victim sensor, and discuss the relevance to real search and rescue robot systems.**

## I. INTRODUCTION

The primary goal of robotic search and rescue is to explore a disaster zone and provide as much information as possible on the location and status of survivors. While the development of advanced and robust robot platforms and systems is essential, high-level problems such as mapping, exploration and multi-agent coordination must be solved as well. Development of such high-level techniques is the goal of RoboCup's Virtual Robots competition. This competition uses USARSim as a basis for its simulations due to this simulator's high quality image data and rendering.

Since the primary goal of robotic search and rescue is finding victims, a simulated robot rescue team must be able to complete this task in simulation. In real rescue systems, identification of victims is often performed by human operators watching camera feedback. To lower the burden on teams using USARSim for rescue systems research, a 'VictimSensor' has been developed for the simulator that mimics recognition of victims in real systems [1]. Modeled after template based human form detection, this sensor must be associated with a camera and performs line-of-sight calculations to the victim. It starts reporting victims at a distance of about 6 metres, and its accuracy improves with increased proximity. However, in this paper we report on work-in-progress towards providing a fast vision-based detector as an alternative, based on the work of Viola and Jones. The hope is that this will provide a more realistic simulation of real-world victim detection, and bring USARSim-related research one step closer to reality.

A secondary goal of search and rescue efforts is to produce high quality maps. One of the reasons to generate a map is to convey information, and this information is often represented as attributes on the map. In addition to victim information, useful maps contain information on the location of obstacles or landmarks, as well as the paths that the individual robots took.

With a view to improving map quality, we have developed a system for the automated labeling of recognisable items in USARSim. In robotics, there is often a need for a system that can locate objects in the environment – we refer to this as 'object detection'. Our detection system exploits the high quality image data from USARSim which allows for accurate classifiers to be trained with a relatively low false positive rate. Using images from USARSim as training data, we have trained various different classifiers to detect various objects, including victims.

The paper is structured as follows. Section II describes related work in object recognition and mapping. Section III provides an overview of our system, including the object detection method used and the training process. In Section IV we detail the process of integrating object detection and mapping into our controller software. In Section V we present preliminary results. Several possible extensions to our object detection systems exist. Some of these are detailed in Section VI followed by concluding remarks in Section VII.

## II. RELATED WORK

Object recognition is a well-studied field of computer vision. Swain and Ballard [2] first proposed colour histograms as an early view-based approach to object recognition. This idea was further developed by Schiele and Crowley [3] who recognised objects using histograms of filtered responses. In [4], Linde and Lindeberg evaluated more complex descriptor combinations, forming histograms of up to 14 dimensions. Although these methods are robust to changes in rotation, position and deformation, they cannot cope with recognition in a cluttered scene.

The issue of where in an image to measure has an impact on the success of object recognition, and thus the need for 'object detection'. Global histograms do not work well for complex scenes. Schneiderman and Kanade [5] were among the first to address object categorisation in natural scenes, by computing histograms of wavelet coefficients over localised object parts. In a similar approach, the popular SIFT (scale-invariant feature transform) descriptor [6] uses position-dependent histograms computed in the neighbourhood of selected image points.

In recent years there has been increasing interest in using object detection in SLAM (simultaneous localisation and mapping) to provide information additional to that provided by laser scans. Such an approach is denoted as visual SLAM (vSLAM). Cameras have an advantage over lasers in that they can offer higher amounts of information and are less expensive. Different methods have been used to extract visual landmarks from camera images. Lemaire and Lacroix [7] use segments as landmarks together with an Extended Kalman Filter-based SLAM approach. Frintrop *et al.* [8] extract regions of interest using the attentional system VOCUS. Others [9] have used SIFT descriptors as landmarks; Se *et al.* [10] and Gil *et al.* [11] track the SIFT features in successive frames to identify the more robust ones; Valls Miro *et al.* [12] use SIFT to map large environments. Davison and Murray [13], and Hygounenc *et al.* [14] use Harris Point detectors as landmarks in monocular SLAM. Finally, Murillo *et al.* [15] propose a localisation method using SURF keypoints.

Jensfelt *et al.* [16] integrate SLAM and object detection into a service robot framework. In their system, the SLAM process is augmented with a histogram based object recognition system that detects specific objects in the environment and puts them in the map generated by the SLAM system. Later the robot is able to assist a human when he/she wants to know where a particular object is. This situation is concerned with the problem of detecting a *specific* object as opposed to a general category of objects.

To the authors' knowledge, little work has been done on integrating object detection techniques into high fidelity simulation applications such as USARSim. For the 2007 Virtual Robot Competition Visser *et al.* [17] used a colour histogram approach for victim detection. A 3D colour histogram is constructed in which discrete probability distributions are learned. Given skin and non-skin histograms based on training sets it is possible to compute the probability that a given colour belongs to the skin and non-skin classes. A drawback of this approach is that in unstructured environments there is no a priori data on the colours present in the environment, which could result in a large number of false positives. In this paper we focus on a more advanced method of detecting general classes of objects in USARSim, and putting them into the environmental map as the exploration effort unfolds.

## III. SYSTEM OVERVIEW

*Viola/Jones Algorithm:* The method we use is based on Viola and Jones' original algorithm for face detection [18], which is the first object detection framework to provide accurate object detection rates in real time. Used in real-time applications, the original detector ran at 15 frames per second on year 2000 hardware; it is therefore suitable for object recognition in robot simulation. We also use this method because it is suitable for detecting objects in complex scenes and under varying lighting conditions, which is typical of robot rescue scenarios.

The method uses a variant of the AdaBoost algorithm for machine learning which generates strong decision tree classifiers from many weak ones. The weak learners are based on features of three kinds, all of which can be individually computed quickly at frame rates. However for a $24\times24$ pixel sub-window there are more than 180,000 potential features. The task of the AdaBoost algorithm is to pick a few hundred of these features and assign weights to each using a set of training images. Object detection is then reduced to computing the weighted sum of the chosen rectangle features and applying a threshold. Thus, although training of the classifiers takes *a lot* of computer time, the resultant classifiers can be run very quickly.

*Cascade of Boosted Classifiers:* We adopt a fast approach used in [19] where we cascade many such detectors, with more complex detectors following simpler ones. Input (an image from a robot's camera) is passed to the first detector which decides true or false (victim or not victim, for example). A false determination halts further computation; otherwise the input is passed along to the next classifier in the cascade. If all classifiers vote true then the input is classified as a true example. A cascade architecture is very efficient because the classifiers with the fewest features are placed at the beginning of the cascade, minimising the total computation time required.

*Training classifiers:* For each classifier, our training set consisted of several thousand $400\times300$ images taken from many different USARSim worlds. In the positive examples (i.e. images containing the object of interest), the object was manually tagged with a bounding box and the location recorded in an annotation file. Ideally each bounding box should have the same scale. In addition to improving classification accuracy, this makes it easier to estimate the real world location of detected objects. For faces we used square bounding boxes.

A major issue with accurate object detection is the effect of different viewpoints on how an object looks. For this reason our training set contained images of objects from many different angles.

Training was carried out over several weeks using the popular open source computer vision library OpenCV[1]. To greatly speed up the process we employed the use of a 528 core SGI-ICE cluster at Oxford University's Supercomputing Centre (OSC), as described in the appendix. In the initial phase separate classifiers were trained for frontal and profile views of heads, both at close range and at further distances. The larger the training set the better; in particular it helped to have a very large number of negative examples. The finished detectors contain a few thousand nodes each, with each classifier having depth of a few tens of nodes. For comparison, classifiers were also then trained for common obstacles found in USARSim environments, such as chairs and potted plants.

We found that the false positive rate can be reduced by using images that were misclassified in one stage as negative examples for successive stages. Our initial face detectors were incorrectly identifying wheels of cars quite often. When we used images of car wheels as negative examples in the training set, the rate of false alarm went down. An example of successful detections, along with one false positive, are shown in Figure 1. (To date the rate of false positives seems acceptable to us for a Robocup Rescue scenario, given that they would presumably then be screened by the operator; however the system is yet to be tested under competition conditions.)

[1]`http://sourceforge.net/projects/opencvlibrary/files/`

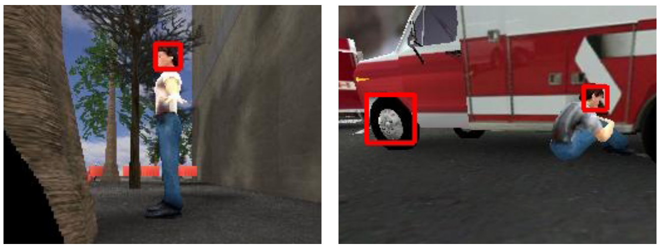

Fig. 1. Our face detector can recognise faces at greater distances than the VictimSensor. Some detections are false positives, but these can be used as negative examples in subsequent levels of training.

## IV. Integration of Object Detection & Mapping

A key competence for a mobile robot is the ability to build a map of the environment from sensor data. For SLAM we use a method developed by Pfingsthorn *et al.* [20], inspired by the manifold data structure [21], which combines grid-based and topological representations. The method produces highly detailed maps without sacrificing scalability. In this section we describe how we augment our map with the location of objects.

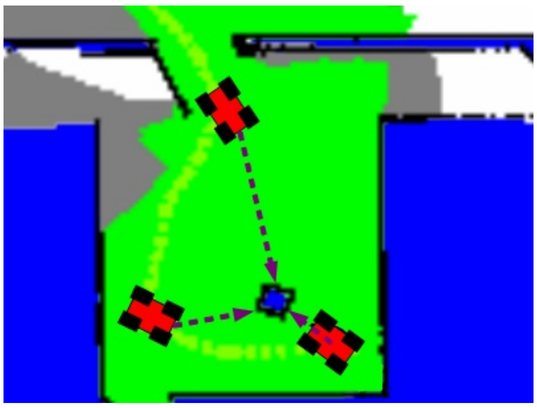

Fig. 2. An object has been detected by the robot from three different locations. The position estimate can be improved by triangulation.

*Finding the position of objects:* Each detector is defined within a single small XML file which is read by a robot's camera sensor; the detector is scanned across the image at multiple scales and locations, using code from the OpenCV library. When an object is detected, the image is saved together with the location of the object in the image and the robot's current pose. The location is taken to be centre of the bounding box. Using the size of an object's bounding box as a gauge of its distance from the robot and its angle relative to the robot, an accurate position estimate is calculated. If the same object is detected from several camera locations the position estimate can be improved by triangulation (Figure 2). This is then placed in the map. An annotated example of a final map is shown in Figure 3.

A position estimate can be quite inaccurate if the bounding box does not fit the object's boundaries closely, since our calculations are based on the real world dimensions of an object.

*Multi-robot object detection:* Using USARSim's Multi-View we can extend our object detection system to multiple robots exploring the environment simultaneously. In this way each robot has its own object detection capability. Ideally the resolution of each subview should be no lower than that of the training images.

*Re-detection of objects:* If a newly detected object is within suitably close range of an existing object already detected, this suggests that it is the same object. The position estimate is made more accurate using triangulation. Re-detection is key to the accuracy of our position estimates. Using triangulation, our position estimates are generally within 1 metre of ground truth. Moreover, re-detection helps us to deal with false positives. Detections occurring only within one frame are likely to be false positives, whereas repeated detections in multiple frames increase the confidence that the detection is correct.

## V. Results

Our object detection system is a work in progress. However, initial results are encouraging, and providing false alarm rates can be reduced, object detection shows promise for use in both high fidelity simulators like USARSim and real robot rescue systems.

*Classification accuracy:* We tested our detector in several standard USARSim worlds, including the `CompWorld` and `Reactor` environments, using multi-robot teams. Figure 4 shows ROC curves for each of our classifiers, and Figure 5 shows some other examples of faces that have been correctly identified by our system, even though they were not detected by the simulated VictimSensor.

Results for faces and plants have detection rates of more than 80% (for some more examples, see Figure 6). However, false alarm rates increase with detection rates. Given our experience now with training for faces on the supercomputer we intend to soon re-visit the issue of training for other objects.

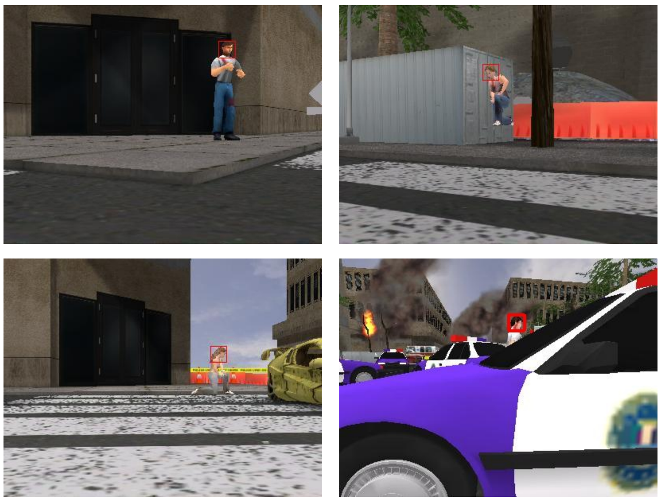

Fig. 5. Four other successful detections of faces.

Our results for hands are less impressive than those for faces and plants. We surmise that there are two reasons for this: firstly, hands come in a wide range of varying poses,

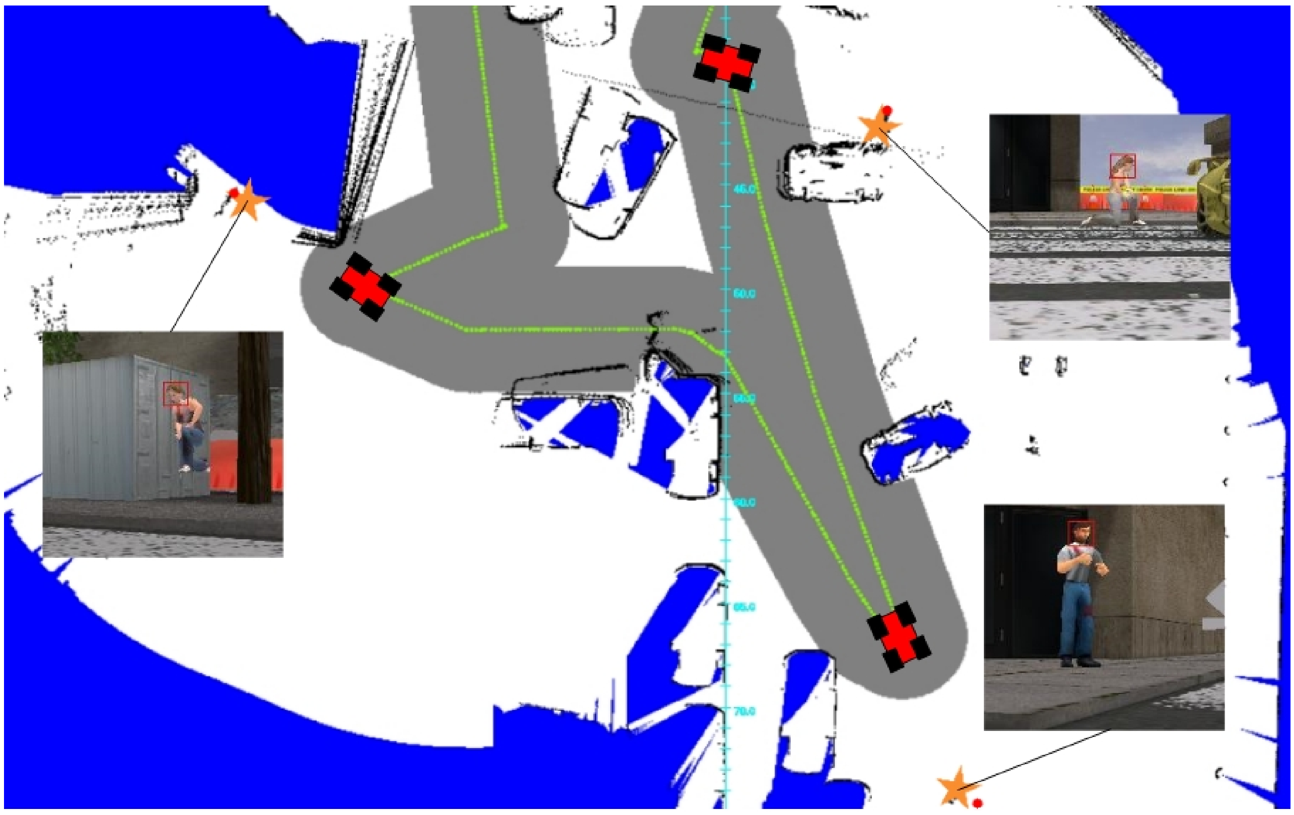

Fig. 3. A completed map together with automatically saved images of detected objects. Green line is the robot's path, orange stars are position estimates and red dots are actual positions.

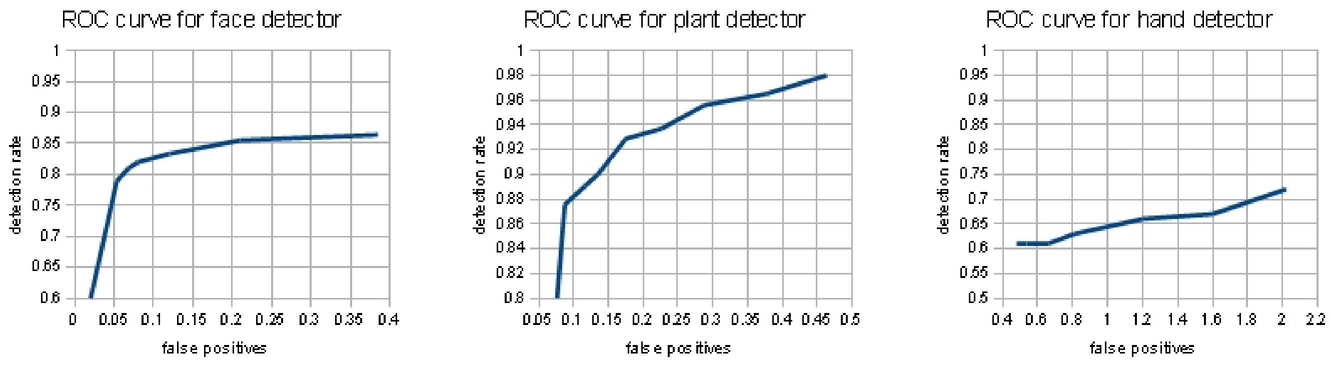

Fig. 4. ROC Curves showing the performance of our object detection system.

from outstretched to clenched fists to hands that are waving. It is therefore difficult to extract the salient features. Faces, conversely, exhibit common features which can easily be learned by a classifier. Plants, particularly the generic potted plants in USARSim, tend to be (vertically) rotation invariant and have the same general characteristics. Secondly, our hand classifier was one of the first classifiers to be trained, before we had access to the supercomputer, so we didn't use as large a training set as would have been optimal.

*Detection time:* To save computation time our detection module searches for objects every $n$ (say, $2 \leq n \leq 4$) frames. The lower $n$ is the more likely an object is to be detected quickly during fast robot motion. Computation time is also dependent on the number of classifiers being passed over a frame. Using an Image Server resolution of 400×300, objects are detected in a few tens of milliseconds on a 2.4 GHz Intel Core 2 processor.

*Real images:* To evaluate the usefulness of our vision-based classifier as a simulation tool, we ran some real images through it. While some faces are classified perfectly, false positive rates significantly increased (for some examples, see Figure 7). This was to be expected, given that real faces exhibit a much higher degree of detail and a much wider variety of features than those used to train our classifier. However, the same classifier has been trained with greater success on real faces by Viola and Jones [19], and the

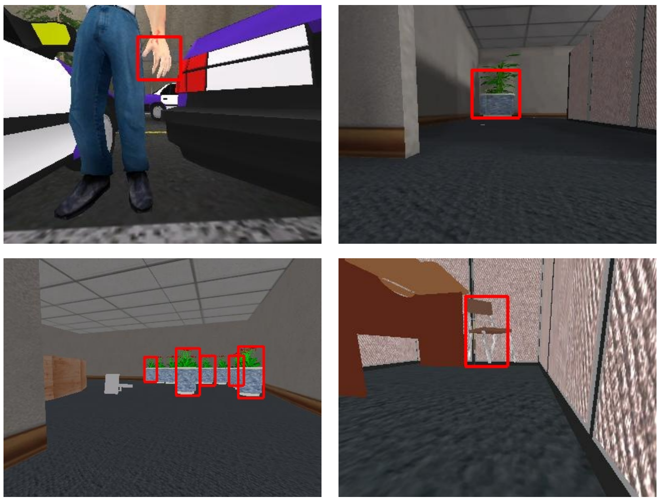

Fig. 6. Examples of other objects detected: hand, plants, chair.

training process for classifiers of real rescue robot systems would not differ from the training process we used within USARSim. In fact, our classifier correctly identifies faces in approximately 88% of simulation images, while Viola and Jones' classifier correctly identifies faces in approximately 90% of real images. Consequently we believe that it is a valid simulation tool: vision-based automated object recognition in real rescue systems would provide similar data and need to be integrated in a similar way to our simulation-based classifier.

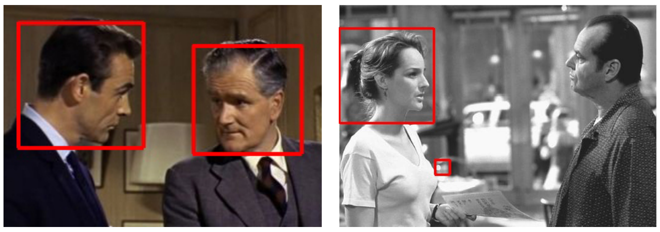

Fig. 7. Results of running real images through our trained classifier. Some faces are recognised well, but the rate of false positives and undetected faces increases significantly.

## VI. FURTHER WORK

Several extensions to our object detection system could lead to further improvements in victim detection and map quality. Some of these are detailed here.

*Improved detection range and detection of more object types:* We hope soon to train classifiers that can detect faces at further distances than at present, using higher resolution images from USARSim. We further hope to train the classifier on a wider range of objects.

*Eliminate need for the simulated VictimSensor:* Currently our face classifiers work for upright faces only. Since the primary goal of robotic search and rescue is to find victims, we plan to extend our victim detection system to victims in differing poses, such as those that are lying down. We hope also to train classifiers for other body parts so as to eliminate reliance on the VictimSensor. If our vision-based victim sensor proves to be very reliable, we envision eventually integrating it either into the simulator itself or into the image server, so that the classifier may be available to the wider USARSim community.

*Tracking by AirRobots:* Since AirRobots are increasingly being used successfully in exploration efforts, most recently in RoboCup 2009 where our team made extensive use of AirRobots in various tests, we plan to use our object recognition system to enable AirRobots to track objects on the ground. This can in turn be used to improve mutual localisation amongst ground robots within the team.

*Object recognition using non-standard views:* A recent addition to USARSim has been the catadioptric omnidirectional camera which provides a full 360 degree view of a robot's surroundings. Additionally, the Kenaf robot platform uses a fish-eye lens to provide a top-down view of the robot. We are interested in investigating whether object recognition can be applied to such non-standard image data using an internal representation of a given object, or using a separate classifier.

*Dust and Smoke:* Real disaster scenes are likely to be subject to dust and smoke; it would be interesting to evaluate our system in the presence of such visual clutter.

## VII. CONCLUSIONS

We have developed a recognition system for faces and common obstacles in disaster zones exploiting USARSim's highly realistic visual rendering. Although the algorithms that we have used are not new, our main contribution is the integration of existing algorithms within a full robot rescue system. One novel feature of our work is the use of a super-computer to train the detectors; without that, the results reported here would not have been achieved.

Our victim detection rivals USARSim's simulated VictimSensor, both in terms of the number of victims found and the distance at which victims may be identified. Detectors for obstacles such as furniture and potted plants allow us to produce richer maps which give a clearer view of an environment's topography. Since our object detectors consist of single XML files generated using open source libraries, they can easily be integrated into any application interface to USARSim.

For chairs and hands the results were less impressive than for faces and plants; chairs are difficult to recognise because their shapes are complex and they are characterised by thin stick-like components, and hands are even more difficult to recognise because there are so many different gestures. However with our recent experience in developing classifiers with the help of the supercomputer cluster we hope to re-visit such items to improve our current classifiers for them.

Our classifier does not perform as impressively on real-world images. This makes sense however, given that it has been trained on simulator images. Similar real-world classifiers for real robot rescue systems could be trained in the same way, as has already been performed by Viola and Jones [19]. Consequently any future research in USARSim that draws conclusions based on a simulated vision-based classifier such as ours is relevant to real-world systems.

In addition, many modern camera systems have the facility to run small pieces of code on the raw image near the camera, and to only then send the results to the main processor. Given the small code size and simplicity of our classifier, it could be run on the camera itself. The object detection results would then be available to any system using the camera. Since this is a realistic possibility in reality, we envision extending USARSim to behave in a similar manner: an object detection system could be built into the image server or the simulator itself, and the detection results would be available to the wider USARSim community.

## VIII. Acknowledgments

The authors gratefully acknowledge the use of the Oxford SuperComputing Centre cluster, without which the classifiers used could not have been generated in a reasonable time.

## References

[1] S. Balakirsky, C. Scrapper, and S. Carpin. The evolution of performance metrics in the robocup rescue virtual robot competition. In *The Evolution of Performance Metrics in the RoboCup Rescue Virtual Robot Competition*, 2007.

[2] M. Swain and D. Ballard. Color indexing. *International Journal of Computer Vision*, Jan 1991.

[3] B. Schiele and J. L. Crowley. Recognition without correspondence using multidimensional receptive field histograms. *International Journal of Computer Vision*, 36:31–50, 2000.

[4] O. Linde and T. Lindeberg. Object recognition using composed receptive field histograms of higher dimensionality. *17th International Conference on Pattern Recognition, 2004*, 2:1 – 6 Vol.2, Jul 2004.

[5] H. Schneiderman and T. Kanade. A statistical method for 3d object detection applied to faces and cars. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1:1746, 2000.

[6] D. Lowe. Object recognition from local scale-invariant features. *International Conference on Computer Vision*, Jan 1999.

[7] T. Lemaire and S. Lacroix. Monocular-vision based slam using line segments. *2007 IEEE International Conference on Robotics and Intelligent Systems*, Jan 2007.

[8] S. Frintrop, P. Jensfelt, and H. Christensen. Attentional landmark selection for visual slam. *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2582 – 2587, Sep 2006.

[9] S. Se, D. Lowe, and J. Little. Vision-based mobile robot localization and mapping using scale-invariant features. *2001 IEEE International Conference on Robotics and Automation*, 2:2051 – 2058 vol.2, Jan 2001.

[10] S. Se, D. Lowe, and J. Little. Global localization using distinctive visual features. *International Conference on Intelligent Robots and Systems*, Jan 2002.

[11] A. Gil, O. Reinoso, W. Burgard, C. Stachniss, and O. Martinez Mozos. Improving data association in rao-blackwellized visual slam. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2076–2081, Beijing, China, 2006.

[12] J. Valls Miro, W. Zhou, and G. Dissanayake. Towards vision based navigation in large indoor environments. *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2096 – 2102, Sep 2006.

[13] A. Davison and D. Murray. Simultaneous localization and mapbuilding using active vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Jan 2002.

[14] E. Hygounenc, I. Jung, P. Soueres, and S. Lacroix. The autonomous blimp project of laas-cnrs: Achievements in flight control and terrain mapping. *International Journal of Robotics Research*, Jan 2004.

[15] A. Murillo, J. Guerrero, and C. Sagues. Surf features for efficient robot localization with omnidirectional images. *2007 IEEE International Conference on Robotics and Automation*, pages 3901 – 3907, Mar 2007.

[16] P. Jensfelt, S. Ekvall, D. Kragic, and D. Aarno. Augmenting slam with object detection in a service robot framework. *15th IEEE International Symposium on Robot and Human Interactive Communication, 2006*, pages 741 – 746, Aug 2006.

[17] A. Visser, B. Slamet, T. Schmits, L. A. Gonzlez Jaime, and A. Ethembabaoglu. Design decisions of the UvA Rescue 2007 team on the challenges of the virtual robot competition. In *Fourth International Workshop on Synthetic Simulation and Robotics to Mitigate Earthquake Disaster*, pages 20–26, Atlanta, July 2007.

[18] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. *2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Jan 2001.

[19] P. Viola and M. Jones. Robust real-time face detection. *International Journal of Computer Vision*, Jan 2004.

[20] M. Pfingsthorn, B. Slamet, and A. Visser. A scalable hybrid multi-robot slam method for highly detailed maps. *Lecture Notes in Computer Science*, Jan 2008.

[21] A. Howard. Multi-robot mapping using manifold representations. *2004 IEEE International Conference on Robotics and Automation*, 4:4198 – 4203 Vol.4, Jan 2004.

## Appendix

### *Training a classifier to detect objects in USARSim*

*Collect images:* A collection of both positive and negative examples is required. Positive examples contain the object of interest, whereas negative examples do not. In the positive examples, rectangles must mark out the object of interest, and these rectangles should be as close as possible to the object's boundaries. Ideally, the bounding boxes in every image should have the same scale.

Four sets of images of required: a positive set containing the object of interest; another positive set for testing purposes; a negative (or 'backgrounds') set for training; and a negative set for testing. The test sets should not contain any images that were used for training. Several thousand images are required, both for positive and negative samples. For frontal faces we used 1500 positive training images, 100 positive testing images, 5000 negative training images and 100 negative testing images.

*Create samples:* OpenCV's `CreateSamples` function can be used to create positive training samples. If there are not sufficient images for training (several thousand are required) additional images may be created by distorting existing images. However, the wider the range of reflections, illuminations and backgrounds, the better the classifier is likely to be trained.

The `CreateSamples` function generates a compressed file which contains all positive images. Assuming a sample size of 20x20 is suitable for most objects, samples are reduced to this size. We experimented with larger sizes, but there was not any noticeable improvement.

*Training:* OpenCV's `HaarTraining` function may be used to train the classifiers. Custom parameters include minimum hit rate, maximum false alarm, type of boosting algorithm and the number of stages. OpenCV developers recommend that at least 20 stages are required for a classifier to be usable. We obtained the best results using 30 stages and the default values for the other parameters.

For our training, we used an Oxford Supercomputing Centre cluster having 66 nodes, each having 8 processors (2 quad core Intel Xeon 2.8GHz) and memory 16GiB DDR2. An essential advantage of this cluster was its parallel processing capability, which allowed for the classifiers to be trained in a reasonable time. Training took approximately 72 hours for each classifier (for comparison, we estimate that the same task on a single PC would take over a month).

*Performance evaluation:* Performance of a classifier can be measured using OpenCV's performance utility. This evaluates the entire testing set and returns the number of correct detections, missed detections and false positives.