Segmentation-Based Online Change Detection for Mobile Robots

Bradford Neuman, Boris Sofman, Anthony Stentz and J. Andrew Bagnell
Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213
{bneuman,bsofman,axs,dbagnell}@ri.cmu.edu

***Abstract*— The high cost of damaging an expensive robot or injuring people or equipment in its environment make even rare failures unacceptable in many mobile robot applications. Often the objects that pose the highest risk for a mobile robot are those that were not present throughout previous successful traversals of an environment. Change detection, a closely related problem to novelty detection, is therefore of high importance to many mobile robotic applications that require a robot to operate repeatedly in the same environment. We present a novel algorithm for performing online change detection based on a previously developed robust online novelty detection system that uses a learned lower-dimensional representation of the feature space to perform measures of similarity. We then further improve this change detection system by incorporating online scene segmentation to better utilize contextual information in the environment. We validate these approaches through extensive experiments onboard a large outdoor mobile robot. Our results show that our approaches are robust to noisy sensor data and moderate registration errors and maintain their performance across diverse natural environments and conditions.**

I. Introduction

As robotics continues to advance, mobile robots are able to navigate more difficult and diverse environments than ever before. In the past several years we have seen robots which can complete complex automated tasks robustly and without major incident, especially in domains such as urban or outdoor navigation [1], [2], [3]. These systems are still not perfect and the chance of failure, however small, discourages many commercial applications for mobile robots. It is becoming increasingly difficult and expensive to continue to improve the performance of these systems and account for all the unknown situations that they may encounter.

Rather than having to operate perfectly in all environments, if a robot can detect situations that are potentially dangerous or outside of its experience base, then it can either avoid those areas entirely or stop and request the aid of a human. For many applications of mobile robotics such as construction, mining, driving supply routes and patrolling, we can leverage the fact that the robot will be operating repeatedly in the same area. If the robot has successfully traversed an environment many times previously, then its biggest sources of risk likely come from aspects of that environment that have changed. The goal of change detection is to robustly discover such changes online as the robot is navigating through a previously traversed scene.

While change detection and the related problem of novelty detection have been well-studied, most algorithms are unsuitable for mobile robotics because they are not suitable for online use and cannot deal with high-dimensional, noisy and redundant features common in robotic perception systems.

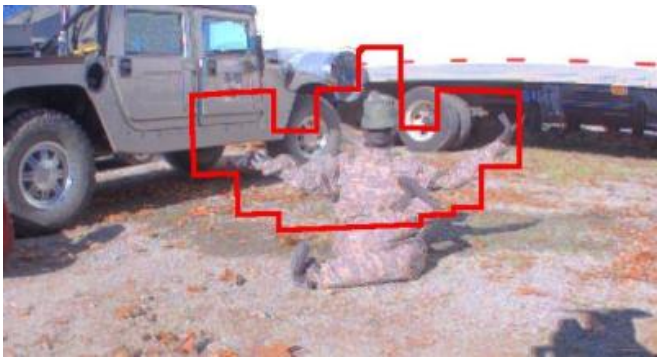


Fig. 1. An example of the system detecting a manikin that was not previously present in this environment.

In this paper we present a change detection system that extends the novelty detection system proposed in [4]. We then show how this system's performance can be dramatically improved through the integration of an online scene segmentation system. This system iteratively segments the scene using a series of Markov Random Field (MRF) optimizations and allows the change detection system to make decisions about entire segments rather than individual 3D locations (we refer to these discretized 3D locations as *voxels*). This improved use of contextual information allows the system to significantly reduce false positives.

The next section discusses related work in novelty and change detection. Sections III and IV present our approach for performing change detection in feature-rich 3D environments and discuss the online scene segmentation system used within. In section V we discuss how these approaches can be applied to the mobile robotics domain and describe the large unmanned ground vehicle (UGV) on which we perform our experiments. Results and analysis are presented in section VI and concluding remarks and future work are in section VII.

II. Related Work

Much previous work has been done in novelty detection (sometimes called anomaly or outlier detection) and applied to various domains. For example, work has been done towards detecting structural faults [5], abnormal jet engine

operation [6], computer system intrusion detection [7], and identifying masses in mammograms [8]. In the robotics domain some have incorporated novelty detection systems within inspection robots [9], [10]. Markou and Singh have written a pair of extensive survey articles detailing many additional novelty detection applications and techniques [11], [12].

Change detection is a much less studied problem but many similar approaches can be used. Several researchers in the computer vision community detect changes in video streams using *temporal difference* methods [13], [14]. Another approach simply compares each current image with the same background at a previous time [15]. These image-based techniques are highly sensitive to varying viewpoints and variations in lighting. Others have applied change detection techniques to analyzing land-cover changes over long periods of time. These include the analysis of forest defoliation [16], reductions of tropical forests [17], and an analysis of land use over time [18]. A survey of various change detection techniques for monitoring land-cover changes can be found in [19].

Most of the existing novelty and change detection techniques are not applicable for mobile robotics because they do not run online, cannot handle noisy and redundant data, or rely on extremely accurate position estimation.

Our change detection system uses online scene segmentation to leverage contextual information in the scene. The problem of segmenting data into meaningful regions has been extensively researched in the computer vision domain [20], [21]. In 3D, people have used graphical methods based on laser scans [22], and have considered splitting scenes into several known classes using feature rich data [23], [24]. These approaches do not work for mobile robots or change detection because they either make unrealistic assumptions or require prior data about scenes they may encounter. Others have used MRFs to segment 3D scenes, but they assume representative training data which cannot be assumed for novelty or change detection [25].

III. Change Detection

In addition to the difficulties discussed in the previous section, change detection is an exceptionally difficult problem for several reasons. First, the system must be able to deal with natural variations in the environment from one traversal to the next, including slight variations in paths driven and resulting variations in sensing angles and density. Such changes would obviously cause variations in computed features between the two traversals that must be filtered to avoid excessive false positives. Also, such techniques must be able to handle small amounts of registration error typical of real-world scenarios. The approach described in this section is able to handle both challenges while maintaining low rates of false positives.

We treat the problem of change detection as a location-specific instance of novelty detection. Rather than trying to identify situations that are novel with respect to everything seen previously, a change detection system needs to identify when a situation is novel with respect to how the situation looked at an earlier time (from a stored log for example).

To accomplish this we leverage the robust online novelty detection algorithm described in [4]. This approach uses a kernel machine to make estimates of similarity where seen examples generate an influence of familiarity in feature space toward future examples. A supervised dimensionality reduction technique, multiple discriminant analysis (MDA), is used to create a lower-dimensional feature space with the property that proximity implies similarity. This approach causes the classes used for training to be projected into clusters that are as compact as possible while being as far away as possible from other cluster centers. In effect, this creates a lower dimensional subspace that truly captures what makes things novel. Novelty is then tested by measuring the accumulated influences of previously encountered examples on the query's location in the feature space.

We use the more compact variation of this algorithm as shown in Algorithm 1. Like in [4], we operate within a lower-dimensional feature space created through supervised dimensionality reduction using labeled classes acquired during the training of the perception system. Since these classes are only required during training this transformation (and not during later use), this approach generalizes well to previously unseen object classes.

Algorithm 1 Online change detection algorithm (modification of algorithm in [4])

1: **given:** A set of voxel features $(\mathbf{x}_i)_{1\ldots N}$ from a location in the current scene; a sequence of corresponding voxel features from a previous navigation $S = (\tilde{\mathbf{x}}_i)_{1\ldots M}$; A function $\mathcal{N}(i, S)$ which finds the neighbors within radius r of $(\mathbf{x}_i)$ in the set S; a novelty threshold γ
2: **outputs:** A sequence of hypotheses $\mathbf{f} = (f(\mathbf{x}_1), f(\mathbf{x}_2), \ldots, f(\mathbf{x}_N))$ where $f(\mathbf{x}_i)$ specifies if voxel $\mathbf{x}_i$ has changed from the previous iteration
3: **for** $i = 1$ to N **do**
4: $\quad y \leftarrow 0$
5: $\quad y \leftarrow \sum_{\tilde{\mathbf{x}} \in \mathcal{N}(i,S)} k(\mathbf{x}_i, \tilde{\mathbf{x}})$
6: $\quad$ **if** $y < \gamma$ **then**
7: $\quad\quad f(\mathbf{x}_i) \leftarrow$ true $\quad$ {This voxel has changed}
8: $\quad$ **else**
9: $\quad\quad f(\mathbf{x}_i) \leftarrow$ false $\quad$ {This voxel is un-changed}
10: $\quad$ **end if**
11: **end for**

We assume a log of all seen voxel features is available from a previous traversal of the environment (only the lower-dimensional representation of the features needs to be stored since comparisons happen in the MDA-based subspace rather than the raw feature space).

As in the original algorithm, we simply use a Gaussian kernel for the kernel function in line 5. For each seen voxel, the kernel function is used to sum the measures of similarity of that voxel with respect to all voxels near that location from the previous traversal. If the example is novel with respect to

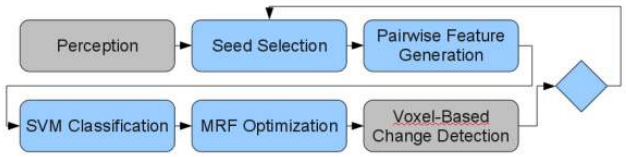


Fig. 2. The control flow through the segmentation-based change detection system. The blue nodes are the steps added by segmentation.

that set of voxels, then the system specifies that the voxel has changed from before. Voxels within some region around the example are considered to allow the algorithm to be robust to moderate amounts of registration error between the two data sets, as well as natural variations in the scene such as vegetation moving in strong wind. Because the addition of any irrelevant voxels provides little contribution toward the novelty threshold in the final prediction (only similar voxels will contribute to the final measure), the addition of these extra voxels does not influence the prediction in a majority of scenarios.

IV. Segmentation System

The approach to change detection described in Section III is analogous to a sliding window approach for image classification. We expand this system to consider contextual information by introducing the online scene segmentation system presented in Algorithm 2. This segmentation system uses a trained general purpose similarity classifier to iteratively segment a scene by constructing and solving a series of two-class MRFs to identify each segment.

Once a segment has been found, the voxel-based change detection algorithm (Algorithm 1) is used to vote on the final classification of all voxels within the segment. This improvement to the change detection system eliminates many false positives by making decisions for regions of the scene instead of individual voxels.

Fig. 2 shows an overview of the entire pipeline for detecting changes using segmentation. Each major step is highlighted in a subsection below, and extended details on the proposed segmentation-based system can be found in [26].

A. Similarity Classifier

In a good segmentation, voxels in the same segment will have a greater similarity than those in different segments. We therefore want to create a classifier that measures similarity between arbitrary voxels (rather than classifying those voxels into any specific classes).

Typically, one would seek a classifier $f(\mathbf{x})$ for a single feature vector. Since we are classifying pairs of voxels, we first apply a feature transformation function $\phi(\mathbf{x}_1, \mathbf{x}_2) \rightarrow \mathbf{x}$ where ϕ is a symmetric mapping of the features of the two voxels into a single feature vector. After extensive experimentation, we converged on using a function which concatenates the magnitude of the difference between the raw features with the sum of the raw features. This allows the classifier to learn based on both the difference between the voxels as well as where in the feature space the voxels were. We will refer to our classifier as $c(\mathbf{x}_1, \mathbf{x}_2) = f(\phi(\mathbf{x}_1, \mathbf{x}_2))$ which is positive if the voxels appear similar (and therefore likely part of the same segment) and negative if they appear dissimilar. The magnitude of the prediction is used as a measure of confidence for that classification.

Training examples for this classifier are created from different-class voxels and adjacent same-class voxels in labeled data collected during perception system training. We use a kernelized support vector machine (SVM) classifier with symmetric Gaussian radial basis functions to efficiently capture the wide range of possible perception inputs[1].

Algorithm 2 Segmentation-based change detection algorithm

given: A set of voxel features $(\mathbf{x}_i)_{1\ldots N}$ from a location in the current scene; a novelty threshold γ; a segment change threshold η
outputs: A sequence of hypotheses $\mathbf{f} = (f(\mathbf{x}_1), f(\mathbf{x}_2), \ldots, f(\mathbf{x}_N))$ where $f(\mathbf{x}_i)$ specifies if voxel $\mathbf{x}_i$ has changed from the previous iteration
while there are any voxels unassigned to a segment **do**
  $s \leftarrow \text{seed}(\mathbf{x})$ {The voxel index selected by the seed selection algorithm (Section IV-B)}
  if $s = -1$ **then** {Segmentation termination criteria reached}
    break
  end if
  Run the similarity classifier between $\mathbf{x}_s$ and every other unlabeled voxel (Section IV-A)
  Create an MRF representing classifier output and neighbor constraints (Section IV-C)
  Solve MRF to get $\mathbf{y}$, the labels for all voxels which minimize the MRF energy function in equation 2
  $\mathcal{S} \leftarrow \{i \ \ \forall i : y_i = 1\}$ {All voxels labeled as part of current segment}
  $\mathbf{g} \leftarrow \Delta(\mathbf{x}, \mathcal{S}, \gamma)$ {Evaluate each voxel in $\mathcal{S}$ independently using Algorithm 1.}
  if $\frac{|\{g_i = 1 \ \ \forall\, i \in \mathcal{S}\}|}{|\mathbf{g}|} \geq \eta$ **then**
    for each i in $\mathbf{g}$ **do** {Entire segment labeled changed}
      $f(\mathbf{x}_i) \leftarrow true$
    end for
  else
    for each i in $\mathbf{g}$ **do** {Entire segment labeled unchanged}
      $f(\mathbf{x}_i) \leftarrow false$
    end for
  end if
end while

[1]For this work, we use the kernelized SVM implementation provided by libsvm[27]

B. Seed Selection and Termination Criteria

We use an iterative segmentation approach in which the first step is to select a seed voxel from which to grow the segment. We want the selected voxel to be a strong representative of one of the remaining distinct segments. Such a voxel needs to correlate strongly with a significant number of remaining voxels. When we discover a segment stemming from this seed, we remove all the voxels belonging to that segment and choose a new seed from the remaining unsegmented voxels.

We select a set of k (12 in our case) random seed candidate voxels $\mathcal{V}$ from the remaining unsegmented voxels and choose the one which maximizes the scoring function:

$$s(x_i) = \sum_{x_j \in \mathcal{V}, i \neq j} \max\{0, c(x_i, x_j)\} \tag{1}$$

This function selects the candidate voxel which is most similar to the other randomly selected voxels (implying they are in the same segment), without regard for those that are likely in different segments.

When the maximum score falls below a threshold (implying that there are no well-defined segments remaining) or there are only a small number of voxels remaining, the algorithm terminates.

C. MRF-based Smoothing

Our similarity classifier gives us segmentation estimates which consider only two voxels at a time. For each pass of the classifier, we have a result representing the similarity between each voxel in the scene and the chosen seed voxel. This approach produces too many false positives because it does not take global smoothness into account. To address this limitation, we use an MRF to represent the constraints from our similarity classifier and our prior smoothness assumption. We solve this MRF by minimizing the energy function:

$$\begin{aligned} E(\mathbf{y}_{1\ldots N}) = & \sum_{i=2}^{N} E_c(\mathbf{x}_1, \mathbf{x}_i) + \\ & \lambda \sum_{\{i,j\} \in \mathcal{N}} \delta(y_i = y_j) + \\ & \infty \delta(y_1 \neq 1) \end{aligned} \tag{2}$$

$$E_c(\mathbf{x}_1, \mathbf{x}_i) = \left\{ \begin{array}{ll} |c(\mathbf{x}_1, \mathbf{x}_i)| & \text{if } \delta(c(\mathbf{x}_1, \mathbf{x}_i) > 0) \neq y_i \\ 0 & \text{otherwise} \end{array} \right\}$$

Where:

- $y_i = 1$ means voxel i is in the same segment as the seed voxel and $y_i = 0$ means voxel i is not in the same segment as the seed voxel. At the start of the optimization, $\mathbf{y}$ contains the result of the similarity classifier.
- $c(\mathbf{x}_i, \mathbf{x}_j)$ is the output of the similarity classifier described in Section IV-A
- λ is the neighbor smoothness weight

The first term in Equation 2 represents the degree to which the segmentation agrees with the evaluations of the classifier. If the sign of the classifier does not match the assignment of the labels, a penalty equal to the magnitude (confidence) of the classifier is applied. The second term applies a neighborhood constraint by penalizing mismatching neighbors by the smoothness parameter λ. Finally, we ensure that the seed voxel $\mathbf{x}_1$ is in segment 1 by applying an infinite penalty if this constraint is not met.

In general, minimizing the energy of a k-class MRFs is NP-hard, but the 2-class problem is a special case that can be solved optimally in polynomial time using a min-cut algorithm if the energy of the distribution of each pair of binary variables is submodular [28], [29].

To solve our MRF represented by the energy function in Equation 2 using the min-cut algorithm, we construct a graph as follows. We turn each remaining voxel in the scene into a node. We next connect nodes corresponding to neighboring voxels with an edge, as in the example in Fig. 3(a). We then create two additional virtual nodes to function as the source and the sink nodes in the graph. These nodes represent the two classes (in-segment and out-of-segment) to be computed by this iteration of the segmentation process. If the classifier says the voxel is the same as the seed, we connect that voxel to the source node with an edge with weight equal to the classifier's confidence. If the classifier predicts that the voxel is different from the seed, we instead connect that voxel to the sink. Finally, the seed node is connected to the source node by an edge with infinite weight, ensuring that it will be part of that class, as seen in Fig. 3(b). Because all edge weights are positive, this graph meets the submodularity requirement for exact 2-class MRF optimization.

After the min-cut optimization is performed, the voxels connected to the source node become the current segment while the voxels connected to the sink node are assumed to be part of other segments and remain for future iterations.

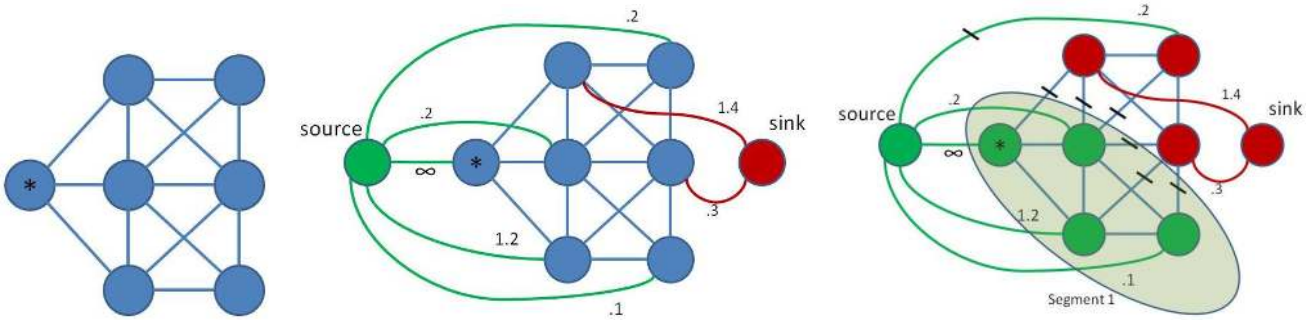
Any edge that is broken in the optimal segmentation is a similarity constraint that was violated. In Fig. 3(c) we see the final segment, circled. We can see that the top right voxel was left out of the segment due to the neighborhood constraint, even though the voxel-classifier supposed (weakly) that the voxel was in the segment.

D. Parameter Optimization

This system has several important parameters which must be properly tuned to achieve good performance.

The similarity classifier SVM (Section IV-A) has two parameters, the slack variable and the kernel bandwidth, which were optimized using a grid search.

The weight on the neighborhood constraint in the MRF, λ (see Equation 2), determines to what extent the MRF optimization can overrule the classifier and enforces the smoothness of the computed segments. A low smoothing parameter would cause the algorithm to under-smooth and increase false positives but would also potentially help in discovering smaller objects, while a large λ will over-smooth the segments and potentially cause the system to miss important changes.



(a) Neighbor constraints (b) Min-cut representation of MRF (c) Final segmented MRF

Fig. 3. MRF optimization process for a sample problem.

The segment change threshold η (Algorithm 2 Line 13) is the fraction of a segment which must be novel in order for the entire segment to be labeled as novel. Low values may cause the algorithm to be overly sensitive to noise within segments while high values will potentially miss important changes, especially when segmentation quality is poor, but will decrease false positives. This parameter is closely coupled to the smoothness parameter because the size of segments effects the ability of the system to isolate changes in the environment. Both parameters can be optimized through gradient-based optimization and cross-validation.

V. Application to Mobile Robotics

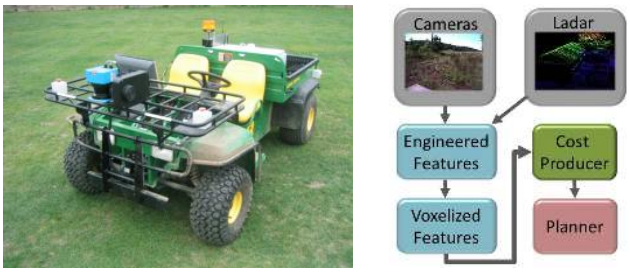


Fig. 4. The E-Gator robotic system (left) and a high-level description of the perception system data flow (right).

An important application of these techniques is to mobile field robotics. We implemented and tested our change detection techniques on the modified John Deere E-Gator shown in Fig. 4. The perception system assigns traversal costs by analyzing the color, position, density, and point cloud distributions of the environment [30], [31]. A large variety of engineered features that could be useful for this task are computed in real-time (see Fig. 5) and the local environment is divided into columns of 20 cm^3 voxels in order to capture all potentially relevant information. Additionally, a real-time ground height estimate is maintained to aid in feature generation and computation [32]. In total, the system generates 30 features for each voxel in the scene. These features are often noisy and redundant and the system is

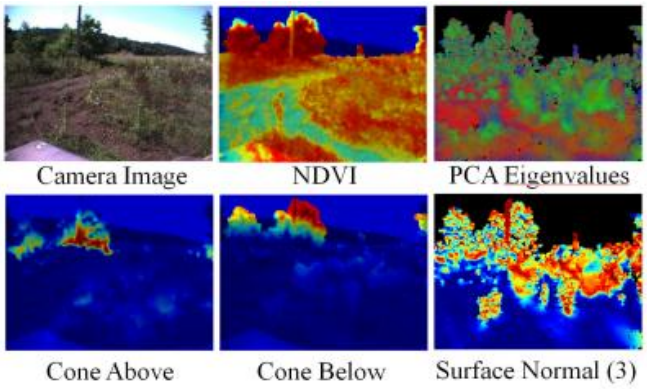


Fig. 5. Example raw and engineered features from the UGV's perception system used by the change detection algorithm. NDVI (normalized difference of vegetation index) is a useful metric for detecting vegetation.

subject to moderate GPS error, so drift of up to a meter is common.

For change detection, we use features from the existing perception system. We train the similarity classifier in Section IV-A with voxels which are labeled with an object class such as "grass" or "building", although we do not need or infer any classes online.

The results presented in the next section are based on experiments performed on the E-Gator UGV system using logged data on similar hardware.

VI. Quantitative Results and Analysis

We do not quantitatively evaluate the direct performance of the segmentation system itself for several reasons. First, unlike in the case of a classification problem, an ideal segmentation is highly subjective and it is impossible to label a ground truth. Also, we use segmentation as an intermediate step in the change detection system pipeline, so for our purposes its performance is better measured by the degree to which it improves the overall systems accuracy.

Change detection performance is evaluated with and without segmentation on pairs of logs from different traversals of the same scene against human-labeled changes. We vary γ (in Algorithm 1) to create an ROC curve representing the system's sensitivity.

We have found experimentally that in most cases incorporating the online segmentation system significantly improves change detection performance. Our total performance across all logs for change detection with segmentation was 84% true positives with 17.76% false positives when we weigh true and false positives equally. To achieve the same true positive rate without segmentation, we would have to incur a 35% false positive rate.

For comparison, we test against two natural occupancy-based methods. We first consider a direct voxel-based occupancy method where we directly compare each voxel against the same 3D location in the previous traversals log and consider a location changed if there was a voxel in one scene and not another. The second method searches for a matching voxel in the previous scene at the same height within a one meter window. As expected, this second approach results in fewer false-positives but misses many changes as a result.

These naive algorithms performed extremely poorly across all logs. This poor performance can be explained by a variety of reasons including extreme vulnerability to registration error and sensitivity to non-rigid obstacles, sensing variations and uncertainty stemming from scene discretization.

Fig. 6 shows the system's performance across all logs. For these ROC results, the system operated at 20% of real-time speed (the perception system runs online at 2 Hz), however there are several performance optimizations that would allow real-time online performance. First, while the change detection system currently runs sequentially after the core perception system, it can run in parallel, decreasing the overall run-time of a perception system iteration. Furthermore, since segments represent disjoint regions of the scene, each segment could be processed for changes in parallel, significantly reducing change detection run-time on a multi-processor system.

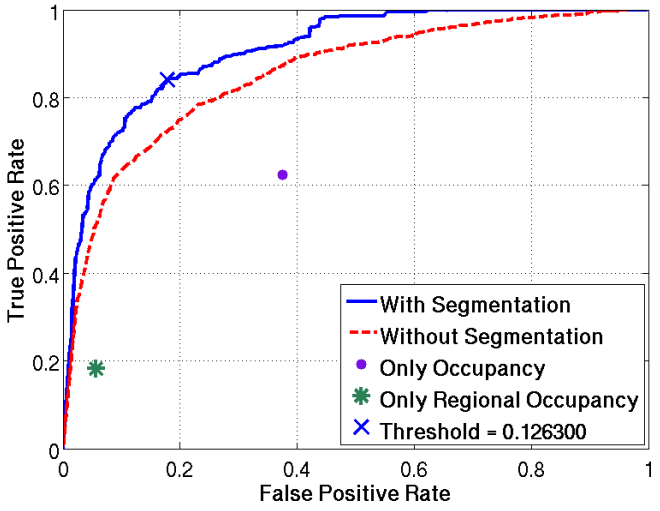


Fig. 6. Performance of the change detection system with and without segmentation

In order to better evaluate the robustness of our system, we tested the system's performance under various amounts of additional registration error (on top of the registration error inherent in the logs). We increased the radius r (in Algorithm 1) to better fit the assumed error model and found that system performance remains stable. Fig. 7 shows this resistance to positioning error by plotting the area under each ROC curve under the various additional errors. Both systems maintain their performance well even under large additional registration error, but the segmentation-based change detection system is clearly more resistant due to its improved ability to eliminate false positives as portions of objects drift begin to mismatch.

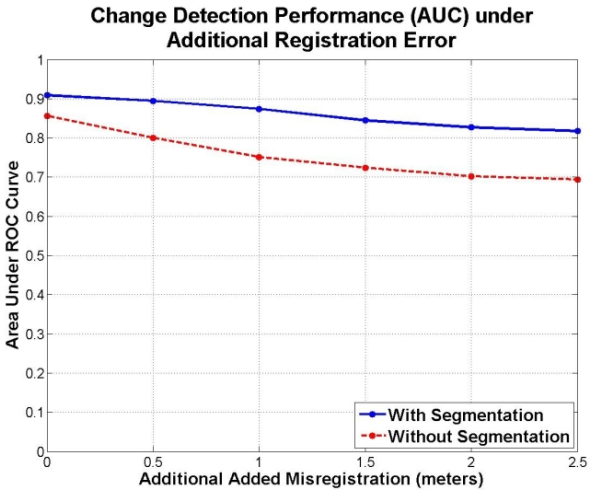


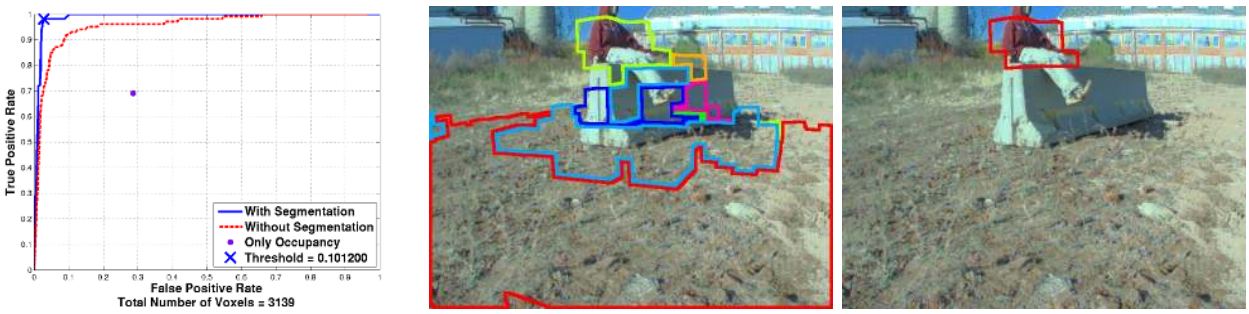
Fig. 7. Area-under-curve for the ROC curves of each approach under additional introduced registration error.

There are several factors that we believe influence the performance of segmentation, and thus change detection. Since we have no a priori assumptions about parameters like number or size of segments, some logs prove very difficult to segment. For example, the log depicting a manikin on a pile of logs in Fig. 8(c) is poorly segmented and results in a poor ROC curve. This may be due to the complexity of the scene and limits to the expressive ability of the perception system. Logs like the one in Fig. 8(a) on the other hand yield excellent segmentation performance and result in high change detection performance.

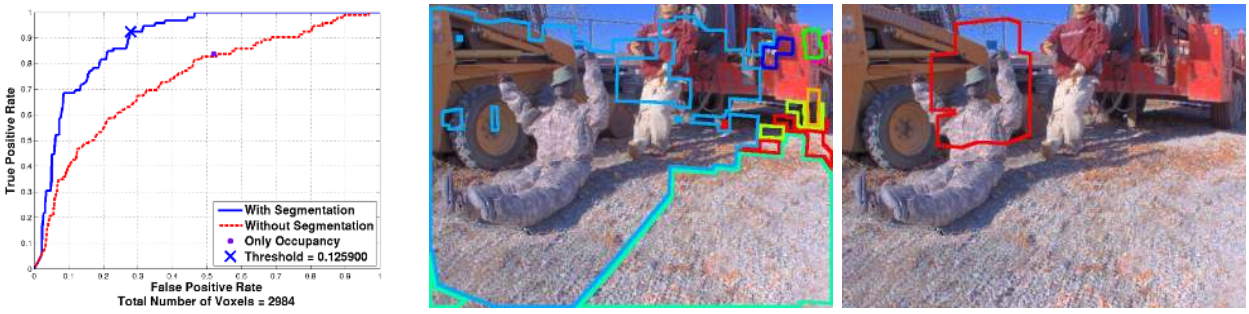
It is important to note that a large source of error also comes from poor human labeling of the change detection pair data logs. When labeling this data it was very difficult to determine exactly which voxels should be marked, especially at the borders of objects. In logs where changes were partially occluded by vegetation, it was extremely difficult to differentiate between voxels belonging to vegetation and those belonging to the added object. The discretization of the scene added further ambiguity.

VII. Conclusions and Future Direction

The techniques presented in this paper provide a way for a mobile robot to detect potentially hazardous changes in its environment. We further demonstrate how this system can be improved through the use of an online scene segmentation system. We demonstrate experimentally the approach's

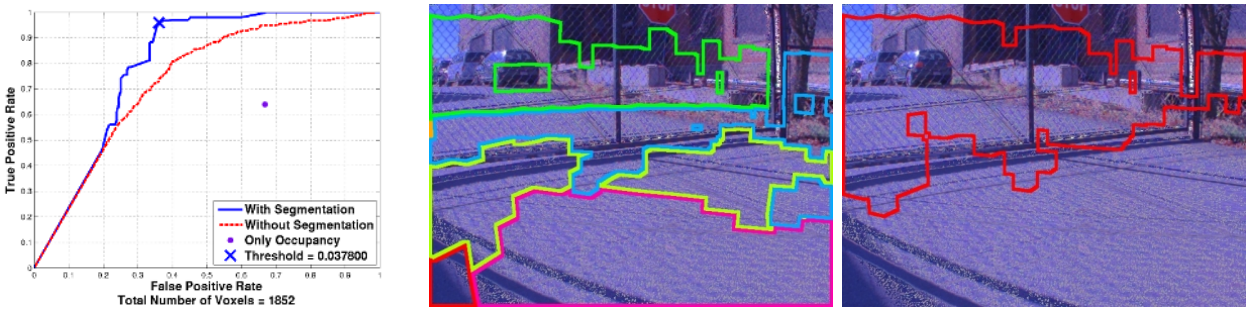


(a) Despite never having trained on an object this shape or color, the system has no problem identifying the manikin as a change.



(b) Despite good ROC performance, this segmentation fails to detect the second manikin entirely.

(c) The segmentation is so poor that none of the segments surpass the segment change threshold η and we see no detected changes at all.



(d) In this log the gate which was previously opened is now closed. The segmentation for the gate is imperfect, but is still helpful. This is a dangerous scenario for a robot because a gate like this may appear traversable due to the low density of the sensor data.

Fig. 8. Performance on selected individual data logs. The ROC performance, scene segmentation, and detected changes are shown from left to right.

ability to run online and deal with localization error and high dimensional, noisy, and redundant data. This work takes a step toward the goal of increasing the fieldability of autonomous mobile robots.

One promising idea to improve performance is using multiple segmentations and combining the results. One could then rank confidence of an entire segment, weighing votes between multiple segmentations with different parameters

and seeds, and combining the best of the segmentations to make the overall change detection more accurate. This idea has been successfully applied in computer vision to determine properties of objects [33], [34].

There are also recent techniques in graphical models which allow the optimization of a graph like our MRF through margin-based approaches. This approach has shown promise in 3D segmentation where the classes are known ahead of time [25], [35].

A key question to consider is how to interact with human supervisors once a change is detected to optimize the human's time. If a human decides that a detected type of change is acceptable, such a change can be handled autonomously in the future. By optimizing humans' time, fewer people can oversee many robots, increasing the efficiency of the entire system.

Another interesting area of future research is an extension of such algorithms to teams of robots. By sharing information it may be possible for groups of robots to detect and handle changes more robustly. This is especially interesting because one of the ideal uses of change detection is in a setting where humans are working with large robot teams.

As robotic systems continue to improve, incorporating the approaches described in this paper into these systems can allow earlier and more effective deployment by acting as a safeguard against the inevitable dangers of unfamiliar domains.

VIII. ACKNOWLEDGMENTS

Bradford Neuman is partially supported by a Graduate Research Fellowship from the National Science Foundation. Boris Sofman is partially supported by a Sandia National Laboratories Excellence in Engineering Fellowship. We would also like to thank Dominic Jonak, Balajee Kannan, Freddie Dias, Jimmy Bourne, Nisarg Kothari, David Silver and other members of the R-CTA Program.

REFERENCES

[1] C. Urmson, *et al.*, "Autonomous driving in urban environments: Boss and the urban challenge," *Journal of Field Robotics*, 2008.

[2] S. Thrun, *et al.*, "Stanley: The robot that won the DARPA grand challenge," *J. Field Robotics*, vol. 23, no. 9, pp. 661–692, 2006.

[3] A. Stentz, J. Bares, T. Pilarski, and D. Stager, "The crusher system for autonomous navigation," in *AUVSIs Unmanned Systems North America*, 2007.

[4] B. Sofman, J. A. Bagnell, and A. Stentz, "Anytime online novelty detection for vehicle safeguarding," in *IEEE International Conference on Robotics and Automation*, May 2010.

[5] K. Worden, "Structural fault detection using a novelty meassure," *Journal of Sound and Vibration*, vol. 201, no. 1, pp. 85–101, 1997.

[6] P. Hayton, B. Schölkopf, L. Tarassenko, *et al.*, "Support vector novelty detection applied to jet engine vibration spectra," in *NIPS*, T. K. Leen, T. G. Dietterich, and V. Tresp, Eds. MIT Press, 2000, pp. 946–952.

[7] J. Ryan, M.-J. Lin, and R. Miikkulainen, "Intrusion detection with neural networks," in *Advances in Neural Information Processing Systems*, M. I. Jordan, M. J. Kearns, and S. A. Solla, Eds., vol. 10. The MIT Press, 1998.

[8] L. Tarassenko, P. Hayton, N. Cerneaz, and M. Brady, "Novelty detection for the identification of masses in mammograms," in *Proceedings of the Fourth International IEEE Conference on Artificial Neural Networks*, vol. 409, 1995, pp. 442–447.

[9] S. Marsland, U. Nehmzow, and J. Shapiro, "On-line novelty detection for autonomous mobile robots," *Robotics and Autonomous Systems*, vol. 51, no. 2-3, pp. 191–206, 2005.

[10] T. Pilarski, J. Bagnell, and A. Stentz, "Hazard detection for familiar terrains via change detection," Master's thesis, Carnegie Mellon University, 2007.

[11] M. Markou and S. Singh, "Novelty detection: a review - part 1: statistical approaches," *Signal Processing*, vol. 83, no. 12, pp. 2481–2497, 2003.

[12] ——, "Novelty detection: a review - part 2: neural network based approaches," *Signal Processing*, vol. 83, no. 12, pp. 2499–2521, 2003.

[13] S. Huwer and H. Niemann, "Adaptive change detection for real-time surveillance applications," in *Third IEEE International Workshop on Visual Surveillance*. Dublin: IEEE, July 2000, pp. 37–45.

[14] N. Paragios and G. Tziritas, "Detection and location of moving objects using deterministic relaxation algorithms," in *International Conference on Pattern Recognition*, 1996, pp. I: 201–205.

[15] Y. Kuno, T. Watanabe, Y. Shimosakoda, and S. Nakagawa, "Automated detection of human for visual surveillance system," in *International Conference on Pattern Recognition*, 1996, pp. III: 865–869.

[16] D. M. Muchoney and B. N. Haack, "Change detection for monitoring forest defoliation," *Photogrammetric Engineering and Remote Sensing*, vol. 60, no. 10, pp. 1243–1251, Oct. 1994.

[17] C. Tottrup, "Forest and land cover mapping in a tropical highland region," *Photogrammetric Engineering and Remote Sensing*, vol. 73, no. 9, pp. 1057–1066, Sept. 2007.

[18] K. Solaimani, S. Modallaldoust, and S. Lotfi, "Investigation of land use changes on soil erosion process using geographical information system," no. 3/603010, 2009.

[19] J.-F. Mas, "Monitoring land-cover changes: A comparison of change detection techniques," 1999.

[20] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient graph-based image segmentation," *International Journal of Computer Vision*, vol. 59, no. 2, pp. 167–181, Sept. 2004.

[21] L. G. Shapiro and G. C. Stockman, *Computer Vision*. Prentice-Hall, 2001.

[22] A. Golovinskiy and T. Funkhouser, "Min-cut based segmentation of point clouds," princeton University.

[23] D. Munoz, N. Vandapel, and M. Hebert, "Onboard contextual classification of 3-d point clouds with learned high-order markov random fields," the Robotics Institute, Carnegie Mellon University.

[24] D. Huber, A. Kapuria, R. R. Donamukkala, and M. Hebert, "Parts-based 3d object classification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR 04)*, June 2004.

[25] D. Anguelov, *et al.*, "Discriminative learning of markov random fields for segmentation of 3D scan data," in *CVPR*, 2005, pp. II: 169–176.

[26] B. Neuman, "Segmentation-based online change detection for mobile robots," Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-10-30, August 2010.

[27] C.-C. Chang and C.-J. Lin, *LIBSVM: a library for support vector machines*, 2001, software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[28] R. Zabih, O. Veksler, and Y. Y. Boykov, "Fast approximate energy minimization via graph cuts," in *ICCV*, 1999, pp. 377–384.

[29] Kolmogorov and Zabih, "What energy functions can be minimized via graph cuts," *IEEETPAMI: IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, 2004.

[30] D. M. Bradley, R. Unnikrishnan, and J. Bagnell, "Vegetation detection for driving in complex environments," in *ICRA*. IEEE, 2007, pp. 503–508.

[31] J.-F. Lalonde, N. Vandapel, D. Huber, and M. Hebert, "Natural terrain classification using three-dimensional ladar data for ground robot mobility," *Journal of Field Robotics*, vol. 23, no. 1, pp. 839 – 861, November 2006.

[32] C. Wellington, A. Courville, and A. Stentz, "Interacting markov random fields for simultaneous terrain modeling and obstacle detection," in *Proc. of Robotics Science and Systems*, June 2005.

[33] T. Malisiewicz and A. A. Efros, "Improving spatial support for objects via multiple segmentations," in *British Machine Vision Conference (BMVC)*, September 2007.

[34] D. Hoiem, A. A. Efros, and M. Hebert, "Geometric context from a single image," in *ICCV*, 2005, pp. I: 654–661.

[35] B. Taskar, V. Chatalbashev, and D. Koller, "Learning associative markov networks," 2004.