2012 IEEE International Conference on Robotics and Automation
RiverCentre, Saint Paul, Minnesota, USA
May 14-18, 2012

Unsupervised Incremental Learning for Long-Term Autonomy

Lionel Ott and Fabio Ramos

Abstract— **We present an approach to automatically learn the visual appearance of an environment in terms of object classes. The procedure is totally unsupervised, incremental, and can be executed in real time. The traversability property of an unseen object is also learnt without human supervision by the interaction between the robot and the environment. An incremental version of affinity propagation, a state-of-the-art clustering procedure, is used to cluster image patches into groups of similar visual appearance. For each of these clusters, we obtain the probability of representing an obstacle through the interaction of the robot with the environment. This information then allows the robot to navigate safely through the environment based solely on visual information. Experimental results show that our method extracts meaningful clusters from the images and learns the appearance of objects efficiently. We show that the approach generalises well to both indoor and outdoor environments and that the amount of learning reduces as the robot explores the environment. This is a fundamental property for autonomous adaptation and long-term autonomy.**

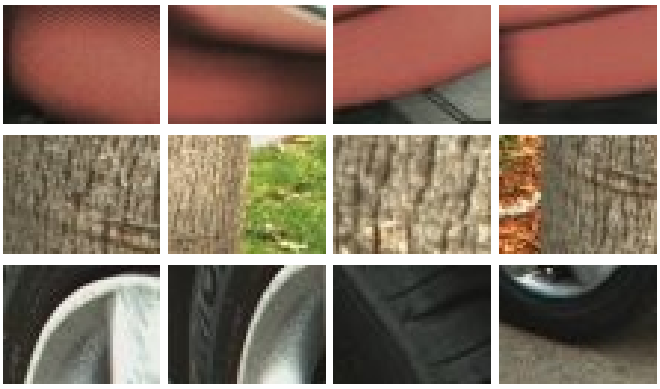


Fig. 1: Examples of the visual appearance of object parts detected by our method. From top to bottom: seat of a chair, a tree trunk and a car tire (best viewed in colour).

I. Introduction

The ability to automatically build a representation of an environment and to adapt to new, unseen scenarios without human supervision is paramount for long-term autonomy in mobile robotics. Additionally, to navigate safely a robot needs to recognise particular properties of objects, the most fundamental one being whether or not an object is traversable. Often such representations and object properties are carefully engineered by a human expert, for example by providing a labelled dataset for learning a semantic map. While this guarantees a representation suitable for the task at hand it can be tedious to obtain and inflexible in case of environment changes. Alternatively, methods that build a model of the environment in an unsupervised fashion have the advantage that they can adapt to changes easily. They also make robots more accessible to non-experts as no special setup is required and can operate in areas where no data is available for prior training.

We propose a method to learn the visual appearance of objects and whether or not they represent an obstacle. The entire learning process runs in real time and without the need for human supervision. We build the model using clustering, specifically, affinity propagation [5]. The advantage of affinity propagation over other clustering methods, such as k-means, is that it does not require the number of clusters to be known a priori. Furthermore, affinity propagation finds high quality clusters while being efficient to compute. An extension of affinity propagation called streaming affinity propagation [21] allows us to cluster data streams in real time. The visual appearance of the environment is encoded by colour histograms and local binary patterns [15]. These features are clustered to generate groups of similar appearance. Interactions of the robot with the environment provides us with information on whether or not an object represents an obstacle. The model built from this information is then used to build a k-nearest neighbour classifier which allows us to label new images into parts that are traversable and non-traversable. Based on the labels obtained from the classifier learnt through self-supervision, a simple decision making procedure can determine safe motion commands for the robot.

Unsupervised learning for robot navigation has received increasing attention in recent years. The methods developed within the DARPA program Learning Applied to Ground Vehicles (LAGR) were concerned with learning to predict terrain traversability from images only. Clustering is often employed in these cases particularly methods that do not require the number of clusters to be defined *a priori* such as spectral clustering [14] and latent Dirichlet allocation [2].

This paper follows the same ideas but it is the first to apply an online version of affinity propagation to the problem. Experiments carried out with a real robot show that the clusters obtained by the proposed method capture the different objects in the environment accurately. We demonstrate that the proposed method improves the recognition of the different objects in the environment as time progresses. Finally, we show that this representation can be used to successfully detect and avoid obstacles in both indoor and outdoor environments.

The main contributions of this work are:

Lionel Ott and Fabio Ramos are with the Australian Centre for Field Robotics, School of Information Technologies, The University of Sydney, Australia.

- a real-time system that learns a model of the appearance of objects in a totally unsupervised manner with streaming affinity propagation;
- a simple and efficient set of visual features to obtain meaningful clusters;
- experimental evidence in both indoor and outdoor environments demonstrating that the robot learns to recognise obstacles in the environment through interaction.

II. Related Work

In the context of the LAGR program interesting methods have been developed that learn to extract traversability information from images observed by a robot. Happold et al. [6] predict the 3D terrain traversability from images based on data obtained from a stereo camera and a neural network classifier. Colour features are linked to geometry and used to predict traversability with a histogram representation. The approach by Howard et al. [7] uses support vector machines to learn the mapping between geometrical features and traversability. This mapping is then used to assign traversability information to clusters of colour features obtained through k-means clustering. The work by Kim et al. [9] uses the experience of the robot as it drives over parts of the environment to train a traversability classifier. A simple incremental clustering method is used to associate the terrain appearance with traversability information by driving over the observed area. Similar to our method these approaches use a combination of colour and texture features to represent the appearance of the environment. However, all but the work by Kim et al. [9] require a supervised classifier to predict traversability, which requires data being labelled by a human expert. The main difference though lies in the way visual appearance is related to the traversability information. Whereas our method uses affinity propagation and thus can infer the number of clusters to use, previous methods either define the number of clusters "a priori" or use simple ad hoc rules for clustering.

Training a classifier with the information gathered by a robot while driving can be used not only to determine traversability, but also for terrain roughness estimation, as shown in the work by Stavens and Thrun [18]. The roughness of the terrain is measured by an inertial measurement unit as the vehicle drives over it. This estimate is then associated with terrain discontinuities extracted from a 3D laser point cloud, thus allowing the vehicle to predict terrain roughness before driving over it; such that it can slow the vehicle down if needed. A similar approach was taken by Ulrich and Nourbakhsh [19] to detect obstacles using only monocular vision. Their method uses the terrain appearance of past trajectories to learn the general appearance of the ground plane. Obstacles are subsequently defined as parts in the image that differ significantly in appearance from the ground plane. The obvious drawback of this method is that it only learns a single model for the ground plane which requires the environment to be uniform. Secondly, training has to be performed by a human driving the robot through the environment, as opposed to our method where the robot learns the model on its own. In a similar spirit Maier et al. [11] use the information of calibrated 3D laser scan and monocular vision to train a ground classifier which is then used to avoid obstacles in absence of continuous 3D data. The work by Modayil and Kuipers [13] is similar to ours in that it collects features from the robot's sensors, a laser scanner in this case, and builds a model of them. While their approach mainly concentrates on the feature extraction and we focus more on the model building.

A method to learn and classify dynamic obstacles in an unsupervised fashion was proposed by Katz et al. [8]. The method uses affinity propagation to cluster laser stamps and visual stamps into groups of detected objects. These are then used in conjunction to classify dynamic objects in the observed scene. As with our approach, affinity propagation would be too slow for the task. They therefore propose a method to incrementally update affinity propagation. The basic idea of their approach is to replace the single exemplar used in affinity propagation with a collection of data points to represent the data set. This effectively reduces the number of points involved in the clustering process.

In the computer vision community, the topic of object detection has been extensively studied and has produced some interesting results. For example, the parts based methods by Weber et al. [20], Agarwal and Roth [1], and Fergus et al. [4] represent an object by a collection of parts from a vocabulary. While these methods successfully learn to detect objects in images it is unclear whether such methods are suitable for robotic applications as the scenes are sterile in comparison to those found in robotic applications. Furthermore the training phase in all the mentioned approaches is too expensive to be performed in real-time.

As indicated by the publications from the LAGR project clustering is an important technique in unsupervised learning. The desired features a clustering method should have are, however, challenging. Most importantly the number of clusters should not be required to be known a priori but determined by the method from the data itself. The work by Kim et al. [9] for example employs a heuristic based clustering method to achieve this. Other more theoretically principled methods include latent Dirichlet allocation [2], spectral clustering [14], DBSCAN [3] and affinity propagation [5]. All of the above methods make different assumptions when modelling the clusters and in the way the clustering is computed. In this work we use affinity propagation due to its simplicity of implementation and flexibility.

III. Unsupervised object discovery

In this section we describe our system which enables a robot to explore the environment and build a representation of it from visual features. The model represents objects present in the environment and if they represent an obstacle. The features are clustered with a combination of affinity propagation and streaming affinity propagation which we review in Section IV. Streaming affinity propagation is responsible for the long-term model of the environment while affinity propagation captures the short-term model. With two

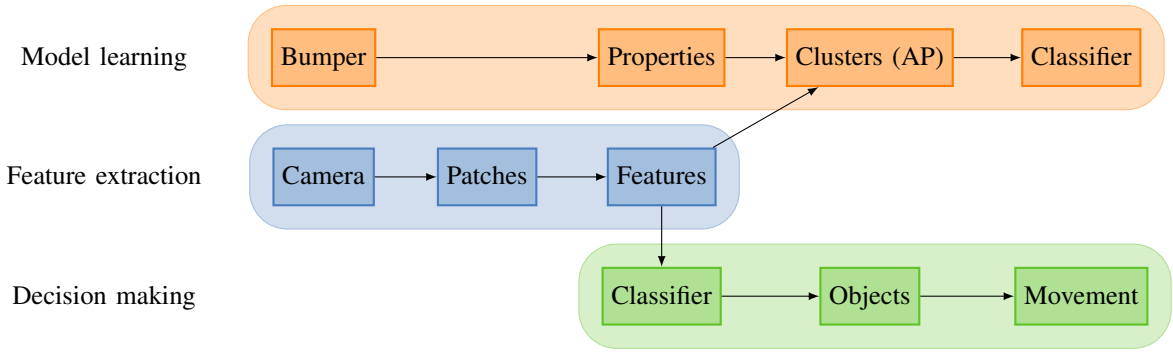


Fig. 2: Overview of the processes of our method. Common to all operations is the division of the image into patches and the extraction of features from those. The learning of a model then proceeds by attaching the obstacle property captured by the bumper to these features. Next the features are clustered using affinity propagation and the result is used to build a classifier. When using an existing model for movement decisions a patch is classified according to its features and the assigned cluster and traversability property is obtained. This information is then used to determine the best motion command for the robot.

separate instances for different time scales we can react quickly to changes in the environment while maintaining a stable long-term model. By continuously adding new observations into the clustering system the model adapts to changes in the environment and improves over time. The labels required for the classification of objects into obstacle and non-obstacle classes are obtained by the robot through collisions, or lack thereof, with the environment. We provide a short overview of the processes involved in the system next.

A. Overview

A schematic overview of the processing pipeline is shown in Figure 2. As a first step our method extracts features from raw images in the following manner (centre row in Figure 2):

1) divide the original image into smaller patches in order to roughly capture a single object per patch;
2) compute colour histograms and histograms of local binary patterns for each of the patches to capture colour and texture information.

Once the features are extracted we can use them to learn a model of the environment as follows (top row in Figure 2):

1) assign each patch a traversability property obtained from the bumper for object classification;
2) add the new features to the clustering system and recompute the clusters to update the model of the environment;
3) use the clustering results to build a k-nearest neighbour classifier to classify new observations as either traversable or non-traversable. A k-nearest neighbour classifier is used as it can be trained quickly from the available data.

With a model of the environment at hand we can make decisions about the motion commands the robot should execute using the following approach (bottom row in Figure 2):

1) obtain the object class for the features extracted from the image patches by classifying them using the k-nearest neighbour classifier;
2) obtain the traversability property associated with each object class;
3) make movement decisions based on the arrangement of traversable and non-traversable parts of the environment.

The steps outlined above will be described in greater detail in the following. We start with the extraction of features and traversability labels. Thereafter, we describe how the model of the environment is built using affinity propagation and how the traversability information is processed. Finally, we show how the learned model can be used determine to motion commands for the robot.

B. Feature Extraction

Images are likely to contain multiple objects with very distinct visual appearances, such as ground, chairs, trees, cars etc. Ideally we would like to compute the features for parts of the image that represent a distinct object. The difficulty is how to select parts of the image that are likely to only contain a single object. We choose the widely used approach of segmenting the image into equally sized rectangular patches. For our application we segment a 320×240 image into 32 rectangular patches of identical size. This has the advantage that it does not require any additional computations while providing a reasonable approximation if the individual patches are small enough. More elaborate approaches, such as watershed based methods [12], might provide better approximations but are also computationally more expensive. From each of the patches we compute two different features: The colour distribution in the HSV colour space and the distribution of local binary patterns [15]. These features allow us to consider both colour and texture when comparing image patches. As we represent the features using histograms, the similarity values needed for affinity propagation are obtained using the Bhattacharyya distance between the colour and texture histograms of pairs of image patches, which is computed as follows:

$$d(H_1, H_2) = \sqrt{1 - \sum_{i=1}^{N} \frac{\sqrt{H_1(i) H_2(i)}}{\sqrt{\sum_{j=1}^{N} H_1(j) \sum_{j=1}^{N} H_2(j)}}}\,, \quad (1)$$

where N is the number of histogram bins, and H_1, H_2 are the two histograms to be compared.

C. Obstacle Label Extraction

In order to associate obstacle information with the learned objects we need to know how the robot interacts with the environment. Whenever the robot collides with obstacles in the environment we assign an "obstacle" label to the currently observed image patches. This information is then transferred to the learned objects represented by clusters. As the robot will never collide with the ground, image patches representing the ground will not be labelled as obstacles, while parts of the environment that represent obstacles, such as walls, chairs, trees and cars will be labelled as obstacles. In the next section we detail how the features and the obstacle labels are used to learn object classes and their obstacle property.

D. Building the Model

Features extracted from image patches are added sequentially into the clustering system. The clustering is performed by affinity propagation (AP) and streaming affinity propagation (STRAP) which are explained in detail in Section IV.

The pseudo code in Algorithm 1 shows the steps performed for each observation we add. Each observation is added to the long-term clustering instance $\mathcal{M}_{long}$ (STRAP), where it is either used to update an existing cluster or added to the outlier reservoir, $outliers$. In the latter case, the data point is additionally added to the short-term clustering instance $\mathcal{M}_{short}$ (AP), which is rebuilt thereafter. When merging the two clustering instances, MERGE($\mathcal{M}_{long}, \mathcal{M}_{short}$), the information about clusters stored in $\mathcal{M}_{short}$ is integrated into $\mathcal{M}_{long}$.

In order to decide if a specific cluster represents an obstacle or not we keep count of how often members of a cluster have been labelled as obstacle and non-obstacle. With these two counts we can easily compute the probability of each cluster representing an obstacle as follows:

$$p(\text{obstacle}_i) = \frac{\#\text{obstacles}_i}{\#\text{obstacles}_i + \#\text{non-obstacles}_i}\,, \qquad (2)$$

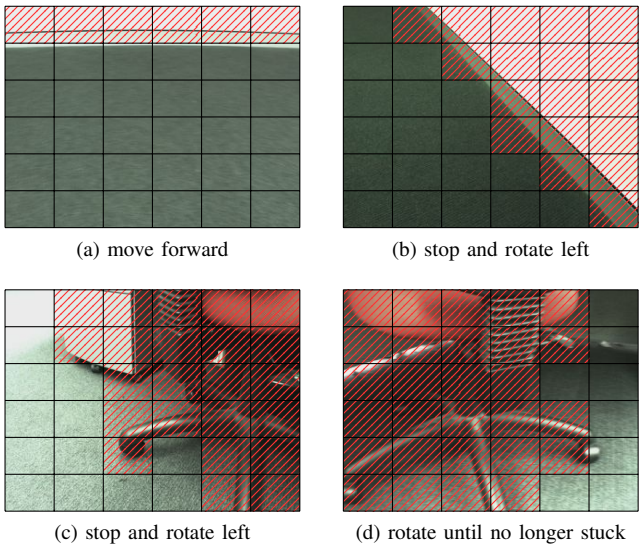
where $p(\text{obstacle}_i)$ is the probability of cluster i representing an obstacle, $\#\text{obstacle}_i$ and $\#\text{non-obstacle}_i$ are the number of image patches in the cluster that were labelled as an obstacle and non-obstacle respectively. This additional information about the clusters is never used in the clustering process itself.

E. Building the Classifier

In order to use the model to predict where obstacles are located in new images a classifier is trained on the exemplars of the clustering. As the method has to run in real-time and the model can change frequently methods that are computationally expensive to train can not be used. For this reason we use a k-nearest neighbour classifier which can be efficiently trained from the clustering result. The training data are the features of the exemplars identified by the clustering, i.e. only a small portion of the original features are used to build the classifier which further reduces the computational cost.

```
ADD-OBSERVATION(z)
1  INSERT(M_long, z)
2  if z ∈ outliers
3      INSERT(M_short, z)
4      UPDATECLUSTERING(M_short)
5  if |outliers| > θ
6      UPDATE-CLUSTERING(M_long)
7      MERGE(M_long, M_short)
8      CLEAR(M_short)
```

Algorithm 1: Pseudo code detailing the steps performed when a new observation z is added to the environment model. $\mathcal{M}_{long}$ is the long-term clustering instance, while $\mathcal{M}_{short}$ is the short-term one.



(a) move forward (b) stop and rotate left (c) stop and rotate left (d) rotate until no longer stuck

Fig. 3: Exemplary classification results and movement decisions. Obstacles are denoted by the shaded areas while the command decision is listed below each image.

F. Decision Making

The first step of the decision making process is to obtain the obstacle information for the patches of the current image, obtained as described in Section III-B. This information is obtained by classifying all patches using the previously trained classifier. This yields an object class and the associated obstacle property for each patch. Using this information, a simple decision making process based on the arrangement of the traversable and non-traversable parts in the image determines safe motion commands for the robot. The procedure checks if there is enough free space in front of the robot to warrant safe forwards movement. Should this be the case the robot is allowed to move forward. Otherwise, the direction which has the best chance to offer free space in front of the robot is determined and an appropriate rotation is executed.

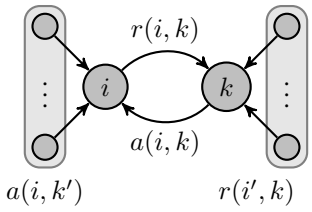


Fig. 4: This figure shows the interaction between the nodes when exchanging messages. The availability messages of the other nodes are used when sending a responsibility message from node i to k. Similarly the responsibilities of all nodes is considered when sending an availability message from node k to i.

Figure 3 shows some images, with obstacles marked by the shaded areas, and the action chosen by the decision making process indicated below each image.

IV. Affinity Propagation

In this section we give a short introduction to affinity propagation [5] and streaming affinity propagation [21], the clustering methods used in our approach. The main advantage of affinity propagation over other popular clustering methods such as k-means, is that it does not require the number of clusters to be defined a priori. This is important since we do not assume any knowledge about the number of objects present in the environment.

A. Optimisation Problem Formulation

Affinity propagation considers the problem of identifying clusters as the search for class label assignments $\mathbf{c} = (c_1, \ldots, c_N)$, called exemplars, that minimises the energy function

$$E(\mathbf{c}) = -\sum_{i=1}^{N} s(i, c_i)\,, \tag{3}$$

where N is the number of data points, c_i is the label assigned to the i-th data point, and $s(i, c_i)$ is the similarity between two data points. The minimisation of the energy function is then reformulated as a maximisation of the net similarity $S(\mathbf{c})$, which is the sum of the negative energy function and a penalty term to enforce valid configurations:

$$S(\mathbf{c}) = -E(\mathbf{c}) + \sum_{k=1}^{N} \delta_k(\mathbf{c}) = \sum_{i=1}^{N} s(i, c_i) + \sum_{k=1}^{N} \delta_k(\mathbf{c})\,, \tag{4}$$

where $\delta_k(\mathbf{c})$ has the form

$$\delta_k(\mathbf{c}) = \begin{cases} -\infty & \text{if } c_k \neq k \text{ but } \exists i : c_i = k \\ 0 & \text{otherwise} \end{cases}\,, \tag{5}$$

and penalises invalid configurations. An invalid configuration is one where a point i chooses another point k as its exemplar without k being labelled as an exemplar. Equation (4) can be solved with loopy belief propagation [16] on the factor graph [10] representation of Eq. (4). A detailed derivation of the messages used in affinity propagation as shown next is given in the supporting online material of [5].

B. Affinity Propagation

The affinity propagation algorithm requires as sole input the similarity values between pairs of data points. From this information a graph is constructed, where the nodes represent the individual data points and edges represent the similarity between pairs of points. The similarity values can, for example, be the Euclidean distance between points or any other similarity measure that is meaningful to the underlying data. While affinity propagation does not require the number of clusters to be defined a priori, it uses the self-similarity values $s(i, i)$ to influence the amount of clusters found.

The computations performed by affinity propagation consist of exchanging two types of messages between connected nodes in the graph. Each of these two message types measures a different property:

- *responsibility* $r(i, k)$, sent from data point i to the candidate exemplar k measures how well-suited data point k is as an exemplar for data point i. This value considers the other potential exemplars for point i as well.
- *availability* $a(i, k)$, sent from the candidate exemplar k to data point i measures how advantageous it would be for point i to choose data point k as its exemplar. This value takes into account the evidence obtained from other data points about the suitability of point k as an exemplar.

These two messages are computed as follows:

$$r(i,k) = s(i,k) - \max_{k' \text{s.t.} k' \neq k} \big(a(i,k') + s(i,k')\big) \tag{6}$$

$$a(i,k) = \min\Big(0, r(k,k) + \sum_{i' \text{s.t.} i' \notin \{i,k\}} \max\big(0, r(i',k)\big)\Big)\,, \tag{7}$$

where $s(i, k)$ is the similarity score between point i and k. The so called self-availability $a(k, k)$ is computed differently:

$$a(k,k) = \sum_{i' \text{s.t.} i' \neq k} \max\big(0, r(i',k)\big)\,. \tag{8}$$

The interaction of these two messages is shown graphically in Figure 4. From this it is visible how sending a responsibility message uses all the current availability messages and vice versa.

To obtain the clustering result the algorithm first initialises all messages to 0 and then iterates the following two steps until convergence (1) update responsibilities (2) update availabilities. Convergence is measured through the net-similarity score of the current clustering which is computed from the responsibility and availability values. Figure 5 shows a sequence of clustering states of a typical clustering run. Starting out with no clear preference until at the end the set of exemplars is found.

C. Streaming Affinity Propagation

While affinity propagation converges reasonably fast, it is not fast enough for use in real-time robotics applications with

(a) State 1 (b) State 2 (c) State 3 (d) State 4 (e) State 5

Fig. 5: Different states in an exemplary run of affinity propagation. The arrows indicate the responsibility message sent from one point to another. Darker arrows indicate a higher message value. At the beginning no point is better suited to be an exemplar then any other, then over time by passing messages the most appropriate exemplars emerge which are marked in red.

a large number of observations. However, there are methods which extend affinity propagation to handle data streams in real time, such as streaming affinity propagation by Zhang et al. [21]. The naïve approach to use affinity propagation for data streaming would be to recompute the clustering for every newly observed data point. This obviously does not work when real-time performance is required. Streaming affinity propagation solves this problem with the following two ideas:

1) reduce the number of data points involved in the affinity propagation computation;
2) limit the number of times affinity propagation needs to be executed.

These two goals are achieved by treating data points as one of two types, those that are similar to existing clusters and those that are dissimilar from the existing clusters. Points that are similar to an existing cluster are used to update the most similar cluster. Points that are dissimilar are added to an outlier reservoir which stores the data points that currently cannot be represented by the clusters. Each cluster is described by a 4-tuple $(e_i, n_i, \Sigma_i, t_i)$ where e_i is the exemplar associated with the cluster, n_i is the number of data points represented by the cluster, Σ_i is the distortion of the cluster, and t_i is the last time a point has been added to the cluster. Once the outlier reservoir is full, affinity propagation is used to recompute the clustering. The similarity values in this case take the information stored in the 4-tuples into account. Once affinity propagation converges, the tuples representing the clusters are recomputed. The net result of this approach is that the affinity propagation algorithm is executed less often and when it runs, the number of data points involved is small.

V. Experiments

In this section we present experimental evaluation results of our method in indoor and outdoor environments. We show results of the clustering quality as well as the learning performance of our system. All the experiments were performed with a Pioneer-AT robot, equipped with a SICK laser scanner and a Point Grey Firewire camera. We used the laser scanner to detect obstacles in close proximity to the robot instead of a bumper in order to avoid damage to both the robot and the environment. The camera on the robot is angled downwards such that obstacles on the floor are visible at a distance of $1.2\,\mathrm{m}$. The images were subdivided into 36 equal sized patches of 52×40 pixels. Other subdivisions were tried but provided similar results.

Our method is implemented in C++ using the Robot Operating System (ROS). All the computations were performed on a Pentium M with $1.7\,\mathrm{GHz}$ at a rate of $5\,\mathrm{Hz}$. The entire process is CPU dependent and only minimal memory is required as only the patches of the exemplars need to be stored for the clustering and classifier.

A. Clustering Quality

We present exemplary results of the cluster centres determined by streaming affinity propagation in Figure 1 and Figure 6. As can be seen the clusters found can be easily distinguished from each other. The exemplars represent the different types of objects found in the environment, i.e. floors, pavement, walls, and predominant obstacles. Ideally clusters should be distinct from each other, i.e. far apart in the feature space. However, clusters should also contain a reasonable amount of data points, i.e. a certain amount of variability within a cluster is required. The examples of cluster members shown in Figure 8 demonstrate that the clusters obtained exhibit this property. Each row in Figure 8 contains members of a single cluster. As can be seen their appearance is sometimes considerably different from each other. Nonetheless they are assigned to the same cluster even though they appear blurred, were observed at a different viewing angle, had different lighting conditions, or were only partially visible. This ability to group similar objects even with diverse appearance allows the overall number of clusters to be kept small and thus more representative of the environment.

For a more accurate evaluation of the clustering quality we compare the clustering results of streaming affinity propagation and k-means using the V-Measure [17]. V-Measure considers the homogeneity and completeness in the computation of the final score. The score is in the range $[0, 1]$ where 1 is the best value. A factor β is used to weight the two measures against each other. In our experiments β was always set to 1, i.e. equal importance is given to homogeneity and completeness. Reference cluster assignments were obtained by labelling multiple sets of image patches by hand. These reference assignments were then used to obtain the V-Measure score for the clustering results of these image sets with streaming affinity propagation and k-means.

Method	V-Measure
Streaming Affinity Propagation	0.763 ± 0.023
k-means	0.726 ± 0.017

TABLE I: Quality of the clustering methods with regards to human reference assignments evaluated using the V-Measure. The mean score of the V-Measure of multiple clusterings as well as the standard deviation is given.

Fig. 6: Examples of the exemplars as determined by streaming affinity propagation. Results from indoor experiments are shown on the left while on the right exemplars obtained in outdoor experiments are shown.

The number of clusters used by k-means clustering was set to the number obtained from affinity propagation. The results in Table I shows that streaming affinity propagation produces higher quality clusters then k-means. A t-test with a significance level of 5% allows us to reject the hypothesis of equal means and thus shows that the increase in V-Measure score of affinity propagation is significant. While this is only a slight increase in accuracy, streaming affinity propagation solves a much harder problem then k-means as it has to determine the number of clusters from the data.

B. Learning Visual Appearance over Time

Our method learns the appearance of objects in the environment by observing them with a camera and labelling them as obstacles when they are very close to the robot (a simulated bump). When the robot starts its exploration there is obviously no information about the environment available. But as time progresses previously unobserved parts of the environment are encountered and their appearance is added to the model.

During this experiment the robot moved in the same area for a period of 15 minutes. The plots in Figure 7 show the percentage of observations made that generate new knowledge as time progresses. This figure shows that the majority of new observations are made early on and that at later stages the number of new discoveries decreases. There are a few instances where the approach perceives previously unseen objects. This can be explained by the robot making a genuinely new observation or observing a previously seen object whose visual appearance is too different to be associated with an existing cluster.

C. Obstacle Avoidance Performance

In order to test the capability of our method to detect and avoid obstacles we placed obstacles in the environment for

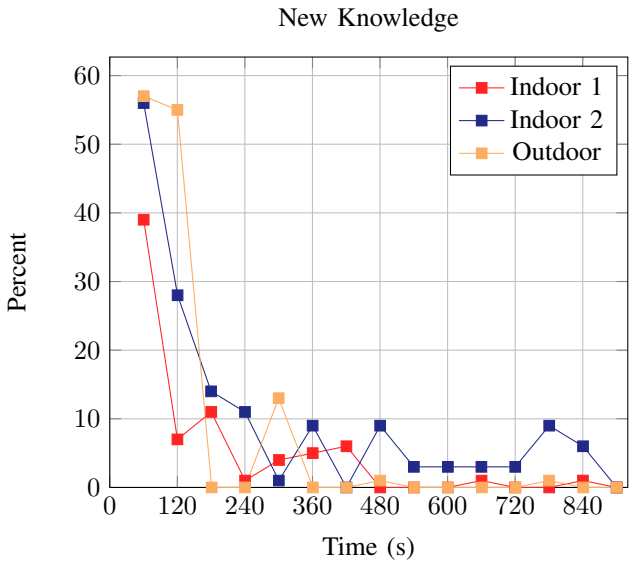


Fig. 7: The plot shows the percentage of all observations made during a 60 s window that add new information to the environment model. It is clearly visible that the majority of observations which lead to new knowledge are observed in the beginning.

the robot to detect. The system was allowed to automatically learn the appearance of these obstacles before the test started. Whenever our method detected an obstacle in front of the robot, the robot stopped and the distance to the nearest obstacle was recorded. This experiment was carried out in different environments. Overall our method recognised and stopped at a distance of $0.94\,\text{m} \pm 0.23\,\text{m}$. This experiment shows that the environment model learned by our approach can be used to detect obstacles well before a collision can occur. This leaves enough time and distance to execute actions to avoid the detected obstacle.

VI. Conclusion

We presented an online method to discover objects in the environment without the need for human supervision. This is achieved by clustering the visual appearance of image patches observed by a robot. Additionally the information obtained by the interaction of the robot with the environment is used to assign the discovered objects with an obstacle property.

Experimental evaluation shows that the proposed system learns and recognises objects which are representative of the environment. Additionally, the obstacle property associated with the objects permit the robot to stop before a collision occurs.

Our method is one of the first attempts to address the problems of long-term autonomy and object discovery with an unsupervised, incremental learning technique. We believe this topic has a lot of potential for future work both algorithmically and experimentally.

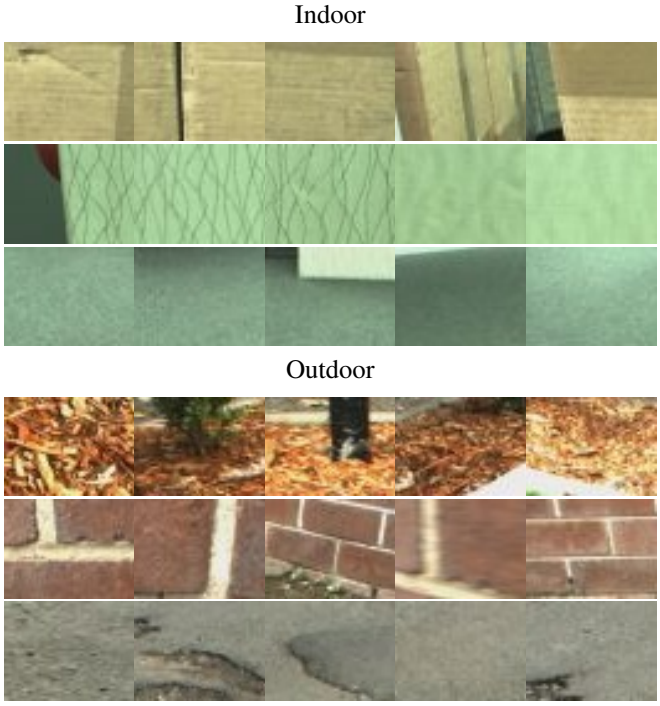


Fig. 8: Examples of cluster members from both indoor and outdoor experiments. Each row contains image patches that are assigned to the same cluster. The examples show that even if the appearance changes significantly between images, the clustering procedure is still able to assign them to the appropriate cluster. The rows from top to bottom represent a cardboard box, a piece of structured room divider, carpet, wood chips, brick wall and asphalt (best viewed in colour).

REFERENCES

[1] S. Agarwal and D. Roth. Learning a Sparse Representation for Object Detection. In *Proc. of the European Conference on Computer Vision (ECCV)*, 2002.

[2] D. Blei, A. Ng, and M. Jordan. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 2003.

[3] M. Ester, H. Kriegel, J. Sander, and X. Xu. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *Proc. of the International Conference on Knowldge Discovery and Data Mining*, 1996.

[4] R. Fergus, P. Perona, and A. Zisserman. Object Class Recognition by Unsupervised Scale-Invariant Learning. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2003.

[5] B.J. Frey and D. Dueck. Clustering by Passing Messages Between Data Points. *Science*, 2007.

[6] M. Happold, M. Ollis, and N. Johnson. Enhancing Supervised Terrain Classification with Predictive Unsupervised Learning. In *Proc. of Robotics: Science and Systems (RSS)*, 2006.

[7] A. Howard, M. Turmon, L. Matthies, B. Tang, A. Angelova, and E. Mjolsness. Towards Learned Traversability for Robot Navigation: From Underfoot to the Far Field. *Journal of Field Robotics*, 2006.

[8] R. Katz, J. Nieto, and E. Nebot. Unsupervised Classification of Dynamic Obstacles in Urban Environments. *Journal of Field Robotics*, 2010.

[9] D. Kim, J. Sun, J.M. Rehg, and A.F. Bobick. Traversability Classification using Unsupervised Online Visual Learning for Outdoor Robot Navigation. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2006.

[10] F.R. Kschischang, B.J. Frey, and H.-A. Loeliger. Factor Graphs and the Sum-Product Algorithm. *IEEE Transactions on Information Theory*, 2001.

[11] D. Maier, M. Bennewitz, and C. Stachniss. Self-supervised Obstacle Detection for Humanoid Navigation Using Monocular Vision and Sparse Laser Data. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2011.

[12] F. Meyer and S. Beucher. Morphological Segmentation. *Journal of Visual Communication and Image Representation*, 1990.

[13] J. Modayil and B. Kuipers. Bootstrap Learning for Object Discovery. 2004.

[14] A. Ng, M. Jordan, and Y. Weiss. On Spectral Clustering: Analysis and an algorithm. In *Proc. of Advances in Neural Information Processing Systems*, 2001.

[15] T. Ojala, M. Pietikainen, and T. Maenpaa. Multiresolution Gray-Scale and Rotation Invariant Texture Classification with Local Binary Patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2002.

[16] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. 1988.

[17] A. Rosenberg and J. Hirschberg. V-Measure: A conditional entropy-based external cluster evaluation measure. In *Proc. of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, 2007.

[18] D. Stavens and S. Thrun. A Self-Supervised Terrain Roughness Estimator for Off-Road Autonomous Driving. In *Proc. of the Conference on Uncertainty in AI*, 2006.

[19] I. Ulrich and I. Nourbakhsh. Appearance-Based Obstacle Detection with Monocular Color Vision. In *Proc. of the National Conference on Artificial Intelligence*, 2000.

[20] M. Weber, M. Welling, and P. Perona. Unsupervised Learning of Models for Recognition. In *Proc. of the European Conference on Computer Vision (ECCV)*, 2000.

[21] X. Zhang, C. Furtlehner, and M. Sebag. Data Streaming with Affinity Propagation. In *Proc. of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, 2008.