Unsupervised online learning for long-term autonomy

Lionel Ott and Fabio Ramos

The International Journal of Robotics Research 2013 32: 1724

DOI: 10.1177/0278364913505657

Article

The International Journal of
Robotics Research
32(14) 1724–1741

DOI: 10.1177/0278364913505657
ijr.sagepub.com

Unsupervised online learning for long-term autonomy

Lionel Ott and Fabio Ramos

Abstract

A reliable representation of the environment a robot operates in is vital for solving complex tasks. Models that represent information about objects and their properties are typically trained beforehand using supervised methods. This requires intensive human labeling which makes it time-consuming and results in models that are generally inflexible to changes. We would prefer a robot that can build a model of the environment autonomously by learning the different objects and their corresponding properties without human supervision. This would enable the robot to adapt to changes in the environment as well as reduce the effort of deploying a robot to a new environment. In this paper we present solutions to these problems based on novel extensions of affinity propagation; a clustering method that can be executed in real time to produce meaningful models from observations gathered by a robot. Our method is applied to two different tasks. We demonstrate how to automatically learn models for predicting collisions from raw laser data. Then, the method is used to learn visual appearance models of the environment to recognize and avoid obstacles. In both cases, there is no human supervision; the methodology is entirely based on sensory information gathered by the robot and its interaction with the environment. In experiments we show how meta-point affinity propagation performs similarly to standard affinity propagation, while being faster and capable of handling much larger data-sets. Furthermore, we show how different features influence the prediction quality of the model for collision prediction from laser scans. Finally, we show how we successfully build and maintain an appearance model for obstacle detection which can be used to detect obstacles well before a collision could occur.

Keywords

Affinity propagation, clustering, long-term autonomy, unsupervised learning

1. Introduction

With robots being deployed in larger and less structured areas, the need for robots to build and maintain reliable models of their environment becomes increasingly important. Typically when a robot is required to identify objects in an environment, a supervised machine learning algorithm, such as support vector machines or boosting, is used. While these methods perform well, there are several drawbacks to them. First, the process of manually annotating the large amount of data required to train the classifier is tedious and error-prone. Second, this approach is inflexible as it is not able to adapt to changes in the environment without more labeling and training. We propose techniques that enable robots to autonomously learn objects and their properties using only their sensors. This has the advantage that no human labor is required and the system can adapt to changes in the environment. It also poses new challenges as learning needs to occur in real time, using limited on-board computational resources.

The main contributions of this paper are:

- An extension to affinity propagation which allows handling of large data streams in real time on board a robot while maintaining good clustering performance;
- A system that learns to recognize and predict collisions with the environment directly from laser scans without the need for human supervision;
- A system that learns and maintains a model of object appearance and their obstacle property in an unsupervised manner.

1.1. Related work

A lot of work has been done in the area of learning terrain properties based on observations. In the LAGR program

Australian Centre for Field Robotics, University of Sydney, Australia

Corresponding author:
Lionel Ott, School of Information Technologies, J12, Australian Centre for Field Robotics, University of Sydney, NSW 2006 Australia.
Email: lott4241@uni.sydney.edu.au

one area of interest was learning terrain traversability based on visual observations by a robot. Happold et al. (2006) link color information to terrain geometry. This geometry information is then turned into a traversability cost using a neural network. A similar approach was taken by Howard et al. (2006) who use a support vector machine to learn the mapping between geometrical features and traversability. This mapping is then used to assign traversability information to clusters of color features obtained through k-means clustering. Kim et al. (2006) use the experience of the robot as it drives over parts of the environment to decide on its traversability property. A simple incremental clustering method is used to associate the terrain appearance with traversability information gained in this fashion directly by the robot. Similar to our proposed method for obstacle learning, these approaches rely on visual features to represent the appearance of the environment. However, all but the work by Kim et al. (2006) require a supervised classifier to predict the traversability of the terrain, which requires data being labeled by a human expert. Another big difference can be found in the way visual appearance is related to traversability information. Whereas our method uses affinity propagation and thus can infer the number of clusters to use from the data, 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 is not limited to determining terrain traversability, but can also be used for terrain roughness estimation, as shown in Stavens and Thrun (2006). 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 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 (2000) to learn 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 main 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. (2011) use the information of a calibrated 3D laser scanner and monocular camera to train a ground classifier which is then used to avoid obstacles using only vision, in the absence of continuous 3D data. The work by Modayil and Kuipers (2004) is similar to ours in that they collect features from the robot's sensors, a laser scanner in their case, and build a model using those. The main difference is that their approach concentrates on the feature extraction and ours focuses on the construction of the model.

Visual appearance can also be used directly without linking to other properties, for example, Giguere et al. (2009) use k-means clustering to learn the model of a coral reef for the purpose of steering a robot such that it remains above the coral reef. The work by Steinberg et al. (2010) uses Dirichlet process mixture models to learn models of the benthic habitats present in image data gathered by an autonomous underwater vehicle. One is not restricted to only visual information either, as the work by Ruhnke et al. (2009) shows: they cluster 3D point clouds using spectral clustering based on a model consistency score.

Katz et al. (2010) propose a method to learn and classify dynamic obstacles in an unsupervised fashion. They build a model by clustering laser stamps and visual stamps using affinity propagation. These clusters are then classified as being either dynamic or static. As with our approach, affinity propagation would be too slow for the task, thus they 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.

Learning to detect objects has also been extensively studied in the computer vision community, which has produced some interesting results. For example, the parts-based methods by Weber et al. (2000), Agarwal and Roth (2002) and Fergus et al. (2003) 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.

Many of the methods mentioned above extract some kind of feature from the sensors available to the robot. These features need to be grouped somehow which often is achieved using clustering methods. For most interesting problems, specifying the number of clusters beforehand is infeasible and we thus need clustering methods able to infer the number of clusters directly from the data. This is non-trivial and in general there is no single correct answer. One solution is to use simple methods based on heuristics, such as in the work by Kim et al. (2006). Another option is to use more theoretically principled methods such as latent Dirichlet allocation (Blei et al., 2003), spectral clustering (Ng et al., 2001), DBSCAN (Ester et al., 1996) and affinity propagation (Frey and Dueck, 2007). All of these methods make different assumptions when modeling the clusters and in the way the clustering is computed. In our work we use affinity propagation due to its simplicity of implementation and flexibility.

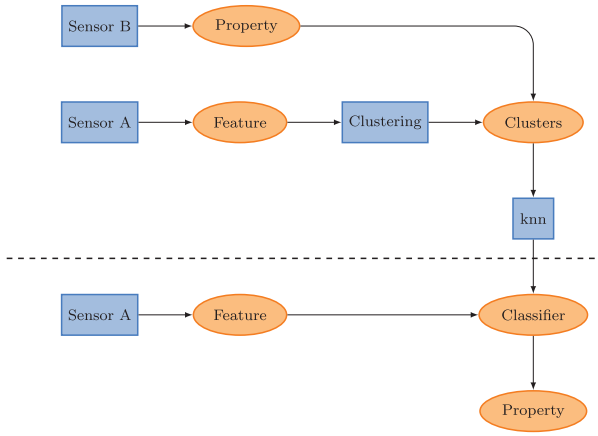


Fig. 1. Processing steps of our pipeline. We build clusters from features extracted from sensor data. These clusters are then associated with properties of the objects in the environment. Using the clusters and the properties, we build a classifier, mapping features to properties. The blue rectangles indicate processing steps producing outputs indicated by the orange ellipses.

1.2. Paper outline

The remainder of the paper is structured as follows. In Section 2 we give an overview of how our systems are structured. Following this, Section 3 introduces affinity propagation and several extensions aimed at improving its computational speed. Section 4 describes our two proposed systems for unsupervised learning of collision detection from laser data and visual obstacle avoidance. Section 5 presents experimental evaluation of the proposed clustering methods and the proposed systems. Finally, Section 6 concludes this paper.

2. Method overview

Our approach to long-term autonomy for robotic systems is based on the idea of unsupervised and self-supervised learning as a means to build and maintain a model of the environment autonomously. This autonomy is crucial as we cannot assume that the environment will remain unchanged over time and thus a model will become inaccurate over time unless it is maintained. Our approach builds the model using features extracted from observations made by the robot. These features are then grouped based on their similarity to obtain clusters representing similar objects in the environment. This provides us with a collection of object appearances present in the environment. However, we often need more information about the objects besides their appearance. We can learn these properties by using the robot's experience from interacting with the environment. If, for example, we are interested in learning which objects are obstacles for the robot we can use the robot's bumper to determine if an observation was made when colliding with an obstacle. This provides us with evidence about which groups of objects represent an obstacle to the robot and which do not. This is performed in a purely self-supervised fashion based on the robot's experience. Combining the model and the properties associated with them allows us to build simple classifiers that enable us to obtain properties of new observations which can be used to make decisions. The different steps of our generic processing pipeline are shown in Figure 1 and explained in greater detail next.

2.1. Feature extraction

The first step of the process is to extract features from the raw data obtained from the sensor. For images this can be SIFT features or color histograms. In the case of range data we can use FLIRT features, line segments or range histograms, for example. These features are then grouped together based on their similarity in the next step.

2.2. Model building

We group the features obtained in the previous step according to their similarities. This is a task well suited to clustering methods, however, the method needs to fulfil several requirements. It must:

1. Determine the number of clusters directly from the data;
2. Be computationally efficient;
3. Be adaptive and incremental.

The first point is important since the method is supposed to work autonomously without any kind of human assistance. The second point is due to the fact that we work with real robots which have limited computational resources at their disposal. Finally, the algorithm should be flexible enough to handle different types of data and adapt to changes. In this work we extend affinity propagation (Frey and Dueck, 2007) to tackle these problems, which are presented in greater detail in Section 3.

2.3. Object properties

The previous step provides a model consisting of a collection of clusters which represents the appearance of objects. Often, however, we need more than just information about appearance. We can obtain further information, or properties, of the learned clusters from the interaction of the robot with the environment. This allows us to perform self-supervised learning of the properties associated with the clusters. For example if we are interested in which clusters represent an obstacle for the robot we can simply monitor which of the clusters are observed when the robot registers a collision. This information is then stored with the clusters and can be used later on.

2.4. Classification

A common use of an environment model is to classify new observations in order to make decisions about the next actions of the robot. With the clusters and their associated

properties we possess enough information to build a simple classifier, such as k-nearest neighbor, to make predictions about the properties associated with new observations. This information can then be used to make decisions about the robot's actions.

3. Affinity propagation

In this section we give a short introduction to affinity propagation (Frey and Dueck, 2007). After that we describe two methods aimed at improving the efficiency of affinity propagation, namely streaming affinity propagation (STRAP) (Zhang et al., 2008), and our own method, meta-point affinity propagation. 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.

3.1. Optimization 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 minimizes the energy function

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

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

$$S(\mathbf{c}) = -E(\mathbf{c}) + \sum_{k=1}^{N} \delta_k(\mathbf{c}) = \sum_{i=1}^{N} s(i, c_i) + \sum_{k=1}^{N} \delta_k(\mathbf{c}) \tag{2}$$

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

$$\delta_k(\mathbf{c}) = \begin{cases} -\infty & \text{if } c_k \neq k \text{ but } \exists i : c_i = k \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

and penalizes invalid configurations. A configuration is considered invalid when a point i chooses another point k as its exemplar without k being labeled as an exemplar. Equation (2) can be solved with loopy belief propagation (Pearl, 1988) on the factor graph (Kschischang et al., 2001) representation of (2). A detailed derivation of the messages used in affinity propagation as shown next is given in the supporting online material of Frey and Dueck (2007).

3.2. Standard affinity propagation

The affinity propagation algorithm requires as its 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 encode 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 message 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:

$$r(i,k) = s(i,k) - \max_{k' \text{s.t.} k' \neq k} \left(a(i,k') + s(i,k') \right) \tag{4}$$

$$a(i,k) = \min \left(0, r(k,k) + \sum_{i' \text{ s.t. } i' \notin \{i,k\}} \max \left(0, r(i',k) \right) \right) \tag{5}$$

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

$$a(k,k) = \sum_{i' \text{ s.t. } i' \neq k} \max \left(0, r(i',k) \right) \tag{6}$$

The interaction of these two messages is shown graphically in Figure 2. 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 initializes 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 3 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.

3.3. STRAP

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

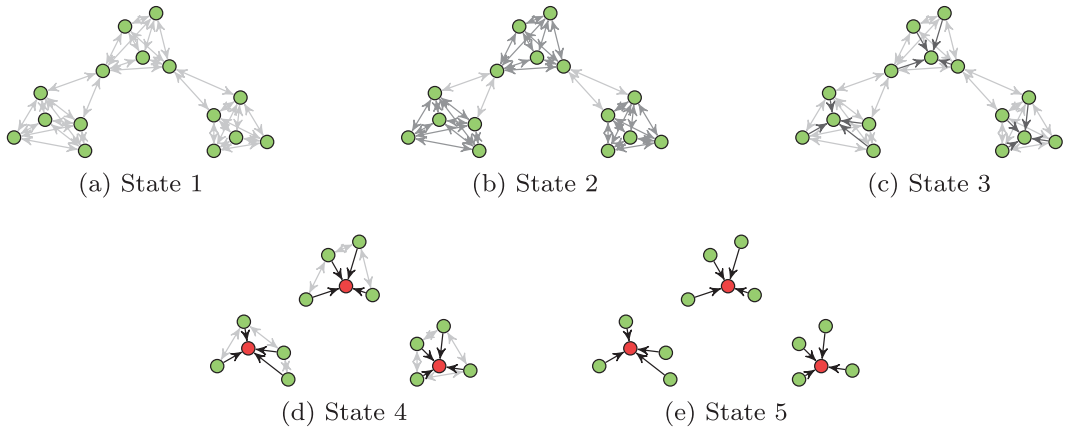


Fig. 3. Different states in an exemplary run of affinity propagation. The arrows indicate the responsibility message sent from one point to another. Darker arrows indicate a higher message value. At the beginning no point is better suited to being an exemplar than any other, then over time by passing messages the most appropriate exemplars, marked in red, emerge.

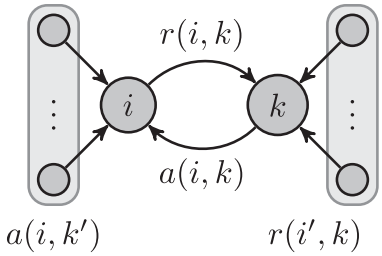


Fig. 2. This figure shows the interaction between the nodes when exchanging messages. The availability messages of neighboring nodes are used when sending a responsibility message from node i to node k. Similarly, the responsibilities of all connected nodes is considered when sending an availability message from node k to node i.

with a large number of observations. However, there are methods which extend affinity propagation to handling data streams in real time, such as STRAP by Zhang et al. (2008). The naïve approach to using 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. STRAP 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 run.

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 was added to the cluster. Once the outlier reservoir is full, affinity propagation is used to recompute the clustering. This requires the computation of similarity values between clusters and the points in the reservoir. These computations take the statistics stored in the clusters into account. The statistics representing the clusters are recomputed once affinity propagation has clustered the data. 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.

3.4. Meta-point affinity propagation

STRAP is fast and capable of clustering data streams, however, there is one drawback stemming from the compression of the clusters. The method runs the risk of 'forgetting' clusters that have not been observed for some time. Depending on the application this may be acceptable. In robotics though this is not the case as we want to be able to recognize areas again even if we have not visited them for some time. For this reason we propose a different method to increase the speed of affinity propagation while guaranteeing that we will not forget information. We therefore propose a method called meta-point affinity propagation, which is inspired by ideas presented in Cao et al. (2006). The main idea is that data points which are close in feature space can be grouped together and replaced by a single meta-point. In robotics, similar observations occur frequently, for example multiple observations made from a similar pose. By replacing such redundant observations with a single aggregated one, we effectively reduce the number of points involved when performing affinity propagation.

A meta-point $\mathbf{P}_i$ stores the following information:

$$\mathbf{P}_i = \{\text{count}, \text{mean}, \text{exemplar}, \text{last-update}\}$$

with the fields having the following meanings:

$\mathbf{P}_i$**.count**	number of points represented by the meta-point
$\mathbf{P}_i$**.mean**	the mean value of all represented data points
$\mathbf{P}_i$**.exemplar**	representative raw data point for this *meta − point*
$\mathbf{P}_i$**.last-update**	time the meta-point was updated last

Besides the immediate effect of reducing the computational burden, the concept of meta-points has two additional benefits:

1. The number of meta-points is dependant on the size of the feature space;
2. Random observations can be dealt with in a straightforward way.

The first point is a direct consequence of the usage of meta-points instead of raw data points. If a robot moves in a static environment all observations will be mapped to one of the meta-points after a while and thus no new meta-points will be created. The second point requires us to distinguish between two types of meta-point: cluster-points that represent the points used for clustering, and noise-points which are ignored during the clustering. A meta-point is considered a cluster-point once it represents enough raw data points, otherwise, it is considered a noise-point. This allows us to discard points generated from random observations such as spurious readings from a laser scanner. Put differently, we can detect and ignore outliers in our observations.

The most important part of meta-point affinity propagation is the handling of new observations. The pseudocode in Algorithm 1 shows the steps performed in order to add a point p into either the set of cluster-points **P** or the set of noise-points **N**. There are three possible cases the algorithm needs to cover:

1. There is a suitable cluster-point present;
2. There is no suitable cluster-point present but only a noise-point;
3. No suitable cluster-point or noise-point is present.

In the first case we simply add the new observation into the meta-point. In the second case we update the meta-point but also check if the noise-point now represents enough data points to be considered a cluster-point. Updating a meta-point is done by recomputing the statistics of the meta-point. Finally, in the third case we create a new meta-point for the observation. These different cases are visualized in Figure 4. This method requires two parameters, $\theta_{\text{min-points}}$, which indicates how many points a meta-point needs to represent for it to be considered a cluster-point, and $\theta_{\text{similarity}}$, which represents the radius of the sphere of influence of a meta-point. The choice of $\theta_{\text{min-points}}$ is mainly dependant on the noise encountered in the data but typically values between 5 and 10 produce good results. However, $\theta_{\text{similarity}}$ is a bit trickier as it is dependant on the magnitude of the feature space. We typically use a value that is 5% to 15% of the magnitude of the feature space.

We use two distinct sets for noise and cluster meta-points to make their difference more obvious, but also for performance reasons. The actual clustering result is obtained by running standard affinity propagation using the cluster-points as input data.

This form of merging data points obviously assumes that small changes in the feature space distance result in no noticeable change of the object class to be clustered. Additionally, the handling of noise only addresses noise which results from random measurements or one-off sensing failures. It does not detect or handle complete failure of a

Algorithm 1: Pseudocode detailing the steps performed by meta-point affinity propagation when a new data point is added. **P** is the set of cluster-points, **N**, the set of noise meta-points and θ, the parameters.

```
ADD-DATA-POINT(p)
1  nn = NEAREST-NEIGHBOR(P, p)
2  if DIST(nn, p) < θ_similarity
3      UPDATE-META-POINT(nn, p)
4  else
5      nn = NEAREST-NEIGHBOR(N, p)
6      if DIST(nn, p) < θ_similarity
7          UPDATE-META-POINT(nn, p)
8          if nn.count ≥ θ_min-points
9              P = P ∪ nn
10             N = N \ nn
11     else
12         noise = CREATE-META-POINT(p)
13         N = N ∪ noise
```

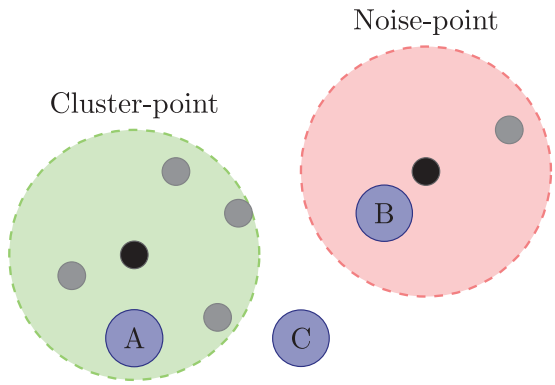


Fig. 4. Visualization of meta-points and the different cases that can occur when adding a new data point. *A* is merged into the cluster-point while *B* is merged into the noise-point. Finally, *C* is used to create an entirely new meta-point.

sensor or systematic noise, as these produce consistent and continuous observations.

4. Self-supervised model learning

4.1. Learning to predict collisions from laser data

In this section we are going to present an approach that enables a robot to learn to predict when it will collide with the environment based solely on laser scanner range information. Our method learns this model without knowing anything about its own physical dimensions or the location of the laser scanner. It is purely based on the laser scanner returns and the information of the bump sensor on the robot. This involves two learning parts, an unsupervised one which learns an appearance model for the laser scans and a self-supervised one which learns the probability of a given cluster to indicate a collision. The combination of these two allows us to learn to predict collisions without having to provide specifics of the robot or the laser scanner to the system.

This system follows the steps outlined in Section 2. As such the first thing we need is a way to compute features from the raw range data. For our case we want a feature that is capable of capturing the broad shape of laser scans. Therefore, we bin the range values equally into several bins. We then compute a single value from all values stored in a bin which yields a histogram for each scan. The following shows the list of functions that can be used to obtain a single value from each bin to produce the histogram representation of a single scan:

Min-value	$\min(\mathcal{S})$
Max-value	$\max(\mathcal{S})$
Mean-value	$\frac{1}{\lvert\mathcal{S}\rvert}\sum_i(\mathcal{S}_i)$
Global difference	$\max(\mathcal{S}) - \min(\mathcal{S})$
Sequential difference	$\frac{1}{\lvert\mathcal{S}\rvert-1}\sum_1^{\lvert\mathcal{S}\rvert-1}(\lvert S_i - S_{i+1}\rvert)$

where $\mathcal{S}$ is the set of the range values contained in a single bin and $\mathcal{S}_i$ is the value of the ith entry in the bin. The first three features capture straightforward statistical information. The last two are a bit more elaborate. The global difference feature tells us how much the distance changes over the entire bin, independent of location. The sequential difference feature gives an intuition on the overall smoothness of the bin and if there are any sudden changes present. Using any of the above simple functions yields a histogram representing a single scan. In order to compute the similarity between the histograms of two scans, needed by affinity propagation, we use the Bhattacharyya distance,

$$d(H_1, H_2) = \sqrt{1 - \sum_{i=1}^{N} \frac{\sqrt{H_1(i)\,H_2(i)}}{\sqrt{\sum_{j=1}^{N} H_1(j) \sum_{j=1}^{N} H_2(j)}}} \quad (7)$$

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

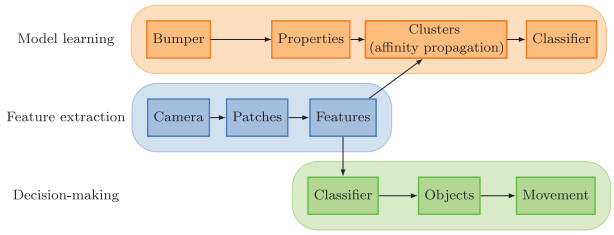


Fig. 5. Overview of the processes of our method. Common to all operations is the division of the image into patches and the extraction of features from those. The learning of a model then proceeds by attaching the obstacle property captured by the bumper to these features. Next the features are clustered 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.

To know which clusters lead to collisions we keep track of the number of observations within a cluster that were made when a collision occurred and the number of times none occurred. With this information we can easily compute a probability of collision for each cluster. In order to know if a new observation is likely to result in a collision we simply select the nearest cluster and obtain its collision probability.

4.2. A model to avoid obstacles

In this section we describe a system which enables a robot to explore the environment and build a representation of it based on visual features. The model represents objects present in the environment and if they are an obstacle to the robot. The features are clustered with a combination of affinity propagation and STRAP. STRAP is responsible for the long-term model of the environment while affinity propagation captures the short-term model. With two separate instances for different time scales we can react to sudden 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 the discovered objects into obstacle and non-obstacle classes are obtained through the robot's experience, that is, collisions or lack thereof with the environment. A short overview of the processes involved in the system is given next.

4.2.1. Overview. A schematic overview of the processing pipeline is shown in Figure 5. As a first step our method extracts features from raw images (center row in Figure 5) in the following manner:

1. Divide the original image into smaller patches in order to roughly capture a single object per patch;

2. Compute color histograms and histograms of local binary patterns for each of the patches to capture color 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 5):

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 neighbor classifier to classify new observations as either traversable or non-traversable.

With a model of the environment at our disposal we can make decisions about the motion commands the robot should execute using the following approach (bottom row in Figure 5):

1. Obtain the object class for the features extracted from the image patches by classifying them using the k-nearest neighbor 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 to determine motion commands for the robot.

4.2.2. Feature extraction. Images are likely to contain multiple objects with very distinct visual appearances, such as the ground, chairs, trees, cars, etc. Ideally we would like to compute the features for parts of the image that represent a single 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. Different numbers of patches have been tried and values anywhere between 9 and 144 worked equally well. However, if the patches get too small the performance will degrade. More elaborate approaches, such as watershed-based methods (Meyer and Beucher, 1990), might provide better approximations but are also more computationally expensive. For each of the patches we compute two different features: the color distribution in the HSV color space and the distribution of local binary patterns (Ojala et al., 2002). These features allow us to consider both color and texture when comparing image patches. As we represent the features using histograms, the similarity values required by affinity propagation are obtained using the Bhattacharyya distance (7).

4.2.3. 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 labeled as obstacles, while parts of the environment that represent obstacles, such as walls, chairs, trees and cars will be labeled 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.

4.2.4. Building the model. Features extracted from image patches are added sequentially into the clustering system. The clustering is performed by affinity propagation and STRAP which are explained in detail in Section 3.

The pseudocode in Algorithm 2 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 $\mathcal{M}_{short}$ (affinity propagation), which is rebuilt thereafter. When merging the two clustering instances, $\text{MERGE}(\mathcal{M}_{long}, \mathcal{M}_{short})$, 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 labeled as obstacle and non-obstacle. With these two counts we can easily compute the probability of each cluster representing an obstacle as follows:

$$p(\text{obstacle}_i) = \frac{\#\text{obstacles}_i}{\#\text{obstacles}_i + \#\text{non-obstacles}_i} \tag{8}$$

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 labeled as an obstacle and a non-obstacle respectively. It is important to note that the clustering is based purely on the visual features and the obstacle properties are never used for this.

There is always the possibility that one visual appearance represents an obstacle in one environment but not in another one. In such a case the model will adapt the obstacle probability of a cluster over time by making new observations. This is not ideal as we try to model two different objects as the same clusters. Therefore, detecting such discrepancies

Algorithm 2: Pseudocode 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.

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

in the model and handling them at a higher level would be preferable.

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

4.2.6. Decision-making. The first step of the decision-making process is to obtain the obstacle information for the current image. As described in Section 4.2.2 this is done 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 and a simple set of heuristics based around the arrangement of the traversable and non-traversable parts in the image we can derive safe motion commands for the robot. The heuristics consider aspects such as the free space in front of the robot and directions most likely to maximize the free space in front of the robot. Figure 6 shows some images, with obstacles marked by the shaded areas, and the action chosen by the decision-making process indicated below each image.

5. Experiments

In this section we present experimental evaluation of our proposed method meta-point affinity propagation as well as its application to laser-scanner-based collision prediction and visual obstacle avoidance learning. Table 1 lists the different experiments we perform as well as the sensors used. First, in Section 5.1, we assess the clustering speed and quality of meta-point affinity propagation on synthetic as well as image data. In Section 5.2 we evaluate the collision prediction performance of the model learned from raw laser scanner data and classification based on bumper feedback. Finally, in Section 5.3 we show the visual appearance model learned from images by our approach as well as the classification of the clusters into obstacle and non-obstacle classes. We also demonstrate how the model is able to deal with long-term operation.

Table 1. Overview of the experiments presented in the following. We list the type of sensor used for the clustering and self-supervision where applicable.

Experiment	Clustering Sensor	Self-Supervision Sensor
Section 5.1.1	Synthetic	None
Section 5.1.2	Camera	None
Section 5.2	Laser scanner	Bumper
Section 5.3	Camera	Bumper (Laser scanner)

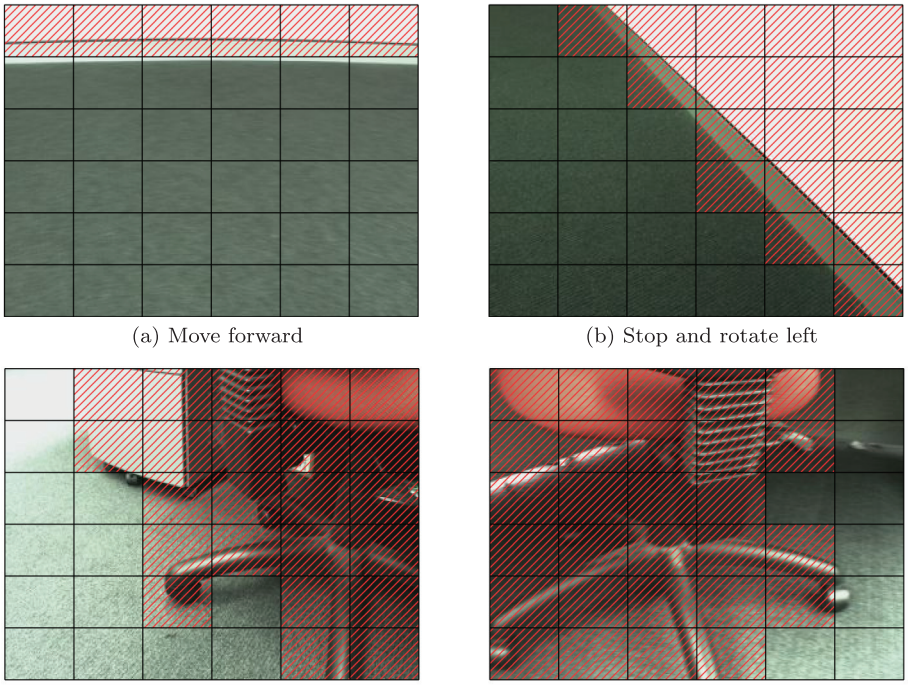
Table 2. Quality of the clustering solutions as computed using V-measure.

Method	V-Measure Score
Affinity propagation	0.87
Meta-point affinity propagation	0.88
STRAP	0.79
k-means	0.84
Mean shift	0.85

5.1. Meta-point affinity propagation evaluation

We evaluate the performance of meta-point affinity propagation as well as the general applicability of the clustering methods to the type of data processed by our systems. The following set of parameters was used for meta-point affinity propagation. In Section 5.1.1, $\theta_{\text{min-points}} = 5$ and $\theta_{\text{similarity}} = 0.2$, and in Section 5.1.2, $\theta_{\text{min-points}} = 5$ and $\theta_{\text{similarity}} = 2.5$.

5.1.1. Clustering quality. Here we will give a comparison of the quality of the clustering results obtained with affinity propagation, STRAP, meta-point affinity propagation, mean shift and k-means using V-measure (Rosenberg and Hirschberg, 2007). V-measure considers the homogeneity and completeness of the clustering solution in the computation of the 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 one, in other words, equal importance is given to homogeneity and to completeness. We cluster a set of 1200 image patches belonging to six different classes: brick wall, asphalt, tree, grass, wood chips and red concrete. Each group is equally represented in the data-set. Exemplary images of these six classes are shown in Figure 7. The number of clusters used by k-means clustering was



(a) Move forward (b) Stop and rotate left

(c) Stop and rotate left (d) Rotate until no longer stuck

Fig. 6. Exemplary classification results and movement decisions. Obstacles are denoted by the shaded areas while the command decision is listed below each image.

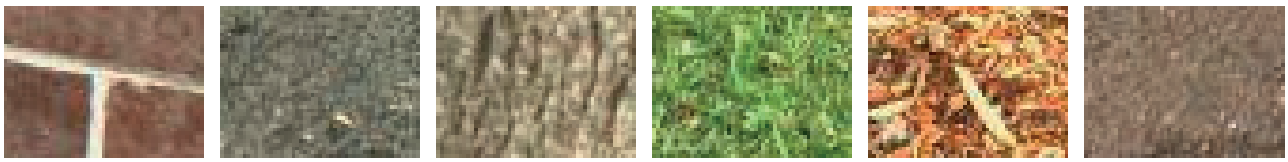


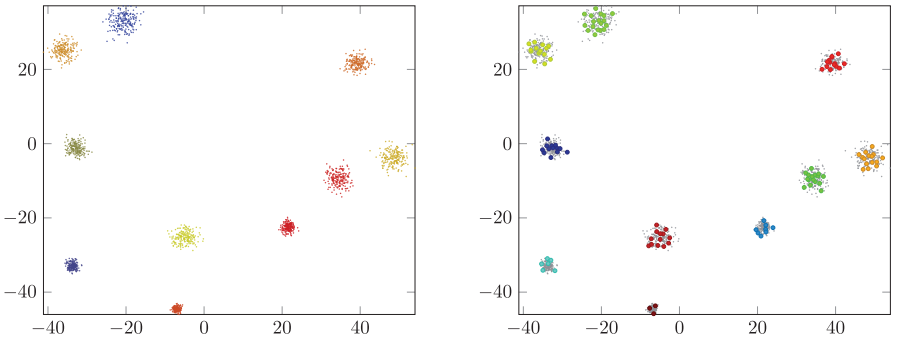
Fig. 7. Examples of image patches contained in the 1200 image patch data-set used to evaluate the clustering performance. From top left to bottom right we have: brick, concrete, tree, grass, wood chips, red surface.

set to six while the other methods were left to find the number from the data itself. The results in Table 2 show that affinity propagation and meta-point affinity propagation perform at a similar level. STRAP, however, performs worse, which can be explained by the way it discards data; k-means performs slightly worse than affinity propagation, but has a simpler problem to solve. Finally, mean shift clustering gives a similar result to k-means. The important thing here is that meta-point affinity propagation and affinity propagation have similar clustering performance, however, meta-point affinity propagation is much more efficient than standard affinity propagation which we show in the following.

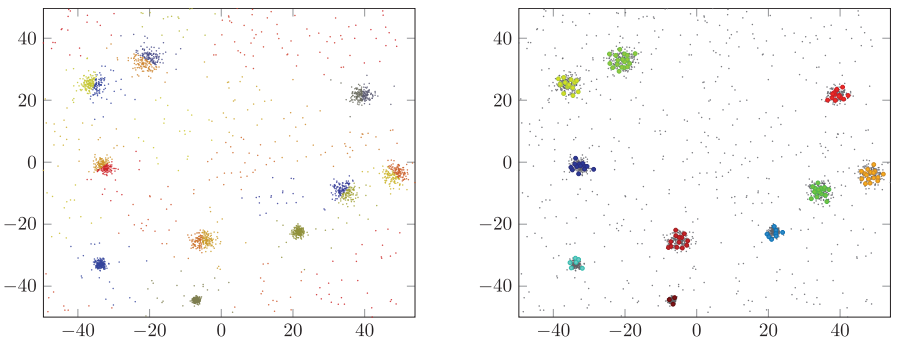
5.1.2. Clustering speed. The previous section showed how both meta-point affinity propagation and affinity propagation produce similar results. Here, we will show differences in speed and the capability of handling noisy data between the two methods. The experiments are performed using synthetic data, shown in Figure 8, as this allows for easier visualization. The data consists of several random 2D Gaussian distributions with optional uniform noise added. In the case without noise both methods find a good clustering solution, as indicated by the colored points. The difference between the two is that affinity propagation has to cluster all the data points, whereas meta-point affinity propagation only has to cluster the meta-points, indicated by the bigger circles. This makes a large difference in speed, which is shown in Table 3. However, in the case where noise is present affinity propagation tends to create a larger number of clusters compared to the result obtained on the same data without noise. This is due to the fact that affinity propagation has to assign each data point to a cluster and cannot consider some observations as noise and ignore them. This often will result in clusters being split, as can be seen in Figure 8(c). Meta-point affinity propagation, on the other hand, first builds meta-points which allows the method to reject points it considers to be noise, and then the clustering has only to deal with data containing a minimal amount of noise. Comparing the meta-points for the data with and without noise we can see that in both cases they cover the actual clusters. Consequently, as far as meta-point affinity propagation is concerned, there is no noise in the data to be clustered.

Table 3. Results for different clustering tasks. Synthetic data shows the results of the 2D example, outdoor data shows the results for the 1200 outdoor image patches data-set and large-scale data shows results for entire sequences captured by a robot moving through the environment. For each of the data-sets we show the number of clusters obtained, the number of raw data points, the number of actually clustered data points clustered and the total run-time. (AP denotes affinity propagation and MPAP denotes meta-point affinity propagation.)

Method	Clusters	Synthetic data		Duration (s)
		Raw points	Clustered points	
AP no noise	10	2000	2000	60.61
AP with noise	35	2500	2500	63.43
MPAP no noise	10	2500	121	0.08
MPAP with noise	10	2500	122	0.08
Method	Clusters	Outdoor Data		Duration (s)
		Raw points	Clustered points	
AP	6	1200	1200	8.79
MPAP	6	1200	79	1.84
Data-set	Clusters	Large-Scale Data: MPAP Only		Duration (s)
		Raw points	Clustered points	
Outdoor	10	31,464	594	185
Indoor	10	48,600	626	307



(a) Affinity propagation, no noise. (b) Meta-point affinity propagation, no noise.



(c) Affinity propagation with noise. (d) Meta-point affinity propagation with noise.

Fig. 8. (a), (b) Exemplary results of affinity propagation and meta-point affinity propagation on data without noise. (c), (d) Exemplary results of affinity propagation and meta-point affinity propagation with uniform noise. The small circles indicate the meta-points found by meta-point affinity propagation. The coloring indicates points that have been assigned to the same cluster.

From the plots of the synthetic data it can easily be seen how meta-point affinity propagation represents the original data with fewer data points. This allows meta-point affinity propagation to be significantly faster than affinity propagation, which is backed up by the data in Table 3. As affinity propagation has quadratic run-time the gaps between the

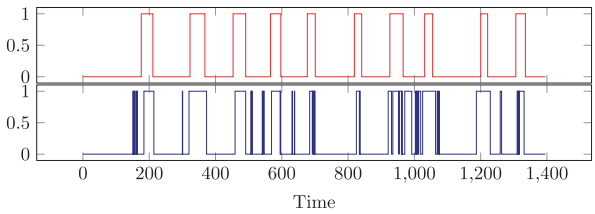


Fig. 9. This figure shows the classification of each single scan as it is received by the robot as it travels through an environment. The top plot shows the ground truth labels and the lower plot shows the prediction made by our method. Values of 0 indicate no collision, while values of 1 indicate a collision.

Table 4. Table showing the area under the curve for the different features and bin sizes evaluated over the two environments. We can see that the min-value, max-value and global difference features perform at a similar level. While the mean-value feature worked well enough indoors it completely fails in the outdoor environment.

	Indoor		Outdoor	
Feature	6 bins	12 bins	6 bins	12 bins
Min-value	0.86	0.89	0.89	0.88
Max-value	0.90	0.90	0.91	0.87
Mean-value	0.87	0.85	0.69	0.81
Global difference	0.90	0.88	0.90	0.65
Sequential difference	0.77	0.78	0.78	0.70

two methods will keep increasing if more points from the same underlying model are added, as meta-point affinity propagation will only update the meta-point statistics, thus keeping the number of points constant. Affinity propagation on the other hand has to handle entirely new points which increases the number of points to cluster. Table 3 also shows the time it takes to process the data used in Section 5.1.1.

To show why meta-point affinity propagation is necessary for use in robotic applications we consider the task of clustering a stream of about 1000 images. Each image is segmented into 36 smaller patches we wish to cluster. This gives us between 30,000 and 40,000 data points to cluster, which is just too much for standard affinity propagation to handle in a reasonable amount of time. However, by using meta-point affinity propagation we can reduce this number to something much more manageable on the order of 500 to 700 points. This means we only need 1% of the original data to obtain meaningful clustering results with meta-point affinity propagation. The 'Large-Scale Data' section in Table 3 shows the run-times for two such data-sets obtained with meta-point affinity propagation.

The results shown here from both synthetic data and real images shows that meta-point affinity propagation obtains results that are comparable with affinity propagation but at a fraction of the computational cost. The added robustness of meta-point affinity propagation to noise makes this new method very appealing for use in robotics.

5.2. Collision prediction from laser scanner data

In this section we evaluate how well our method proposed in Section 4.1 performs the task of grouping laser scans and predicting if they correspond to the robot colliding with the environment. To this end we collected data-sets in which the robot collides with the environment. The data was collected in both indoor and outdoor environments. The main difference between those is that the indoor environment has smaller distances between objects and more reflective objects due to the presence of chairs and tables. The outdoor environment features more areas where no measurements are available due to the limited range of the laser scanner. This data is processed by our method which allows it to learn a model of the laser scans by clustering them and to build collision statistics for each of the clusters. In Figure 11 we show scans collected in an indoor environment. The plots show the scan both in polar and Cartesian coordinates as well as the image taken at the same time. From the plots one can see that there are distinct differences between collision and collision-free scans, in particular the amount of maximum-range readings present in the case of collision observations. These are due to the physical location of the scanner on the robot. The Cartesian plots also show that just relying on those would in many cases make it very hard to decide if a robot is in collision with the environment or not.

In Section 4.1 we proposed several simple functions to obtain a histogram feature from a single laser scan. We are going to evaluate these features with two different bin sizes: six bins with 30 values each and 12 bins with 15 values each. Figure 10 shows ROC curves of the classification performance of the different feature and bin-size combinations. We can see that all plots share an overall trend as far as feature performance goes, with slightly better performance when using only six bins. As for the features themselves, the min-value feature performs best, but only if we accept a large percentage of false positives. Both max-value and global difference perform close to the min-value feature but attain a good and stable performance at a much lower false positive rate. The reason that max-feature performs so well can be explained by the large number of maximum-range readings produced by the laser scanner as shown in Figure 11. Therefore the global difference feature seems the best choice due to its more generic nature. The mean-value feature performs reasonably well indoors but does a poor job outdoors. The sequential difference feature performs similarly in all cases but is outperformed by the global difference feature, which seems to indicate that smoothness is not particularly typical for either collision or collision-free scans. The area-under-curve values corresponding to the ROC curve plots are shown in Table 4 and paint a similar picture. It is easy to see how similarly the min-value, max-value and global difference features perform. Combining this with the data from the ROC curves we can conclude that the simple global difference feature is a good and robust feature to use for this task.

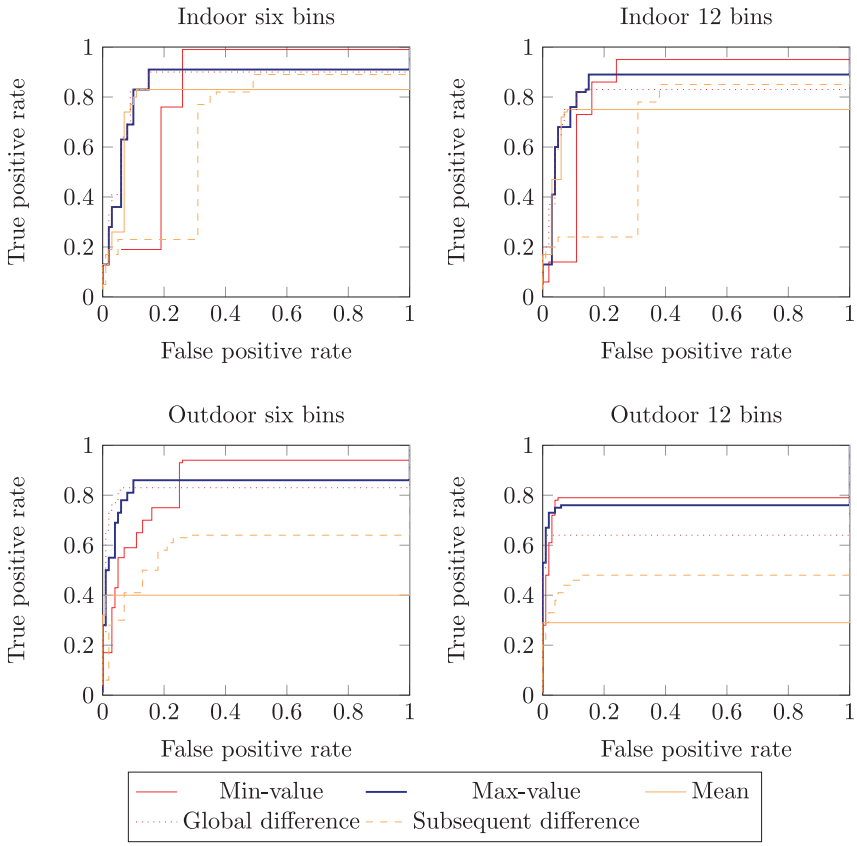


Fig. 10. ROC plot for different features evaluated on different binning sizes of laser scans in indoor and outdoor environments. We can see similar trends independent of the number of bins we use. The "minvalue" feature obtains the best results, however, at a larger false positive rate then other features. Both the "max-value" and "over all difference" features obtain similar results.

Another way to look at how well the learned model performs is by comparing the timeline of collision and non-collision scans. Figure 9 shows the plot obtained for the indoor data-set with the global-distance feature using a threshold of 0.2. The top plot in red shows the hand-labeled ground truth while the bottom plot in blue shows the prediction of our model. It shows that the blue plot mostly covers the actual instances of collisions with some short instances misclassified as a collision. It is worth noting that in many cases the prediction triggers before the actual classification in the ground truth plot, indicating that we could stop the robot before it collides with the environment.

5.3. Visual appearance learning and obstacle avoidance

In this section we present evaluation results of our system, presented in Section 4.2, which learns visual object appearance and obstacle property in both 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. The laser scanner was used to detect obstacles in close proximity to the robot and issue a 'bump' event 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 m. The images were subdivided into 36 equally 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 GHz at a rate of 5 Hz. The entire process is CPU-bound and only minimal memory is required as only the patches of the exemplars are stored for the clustering and classifier.

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

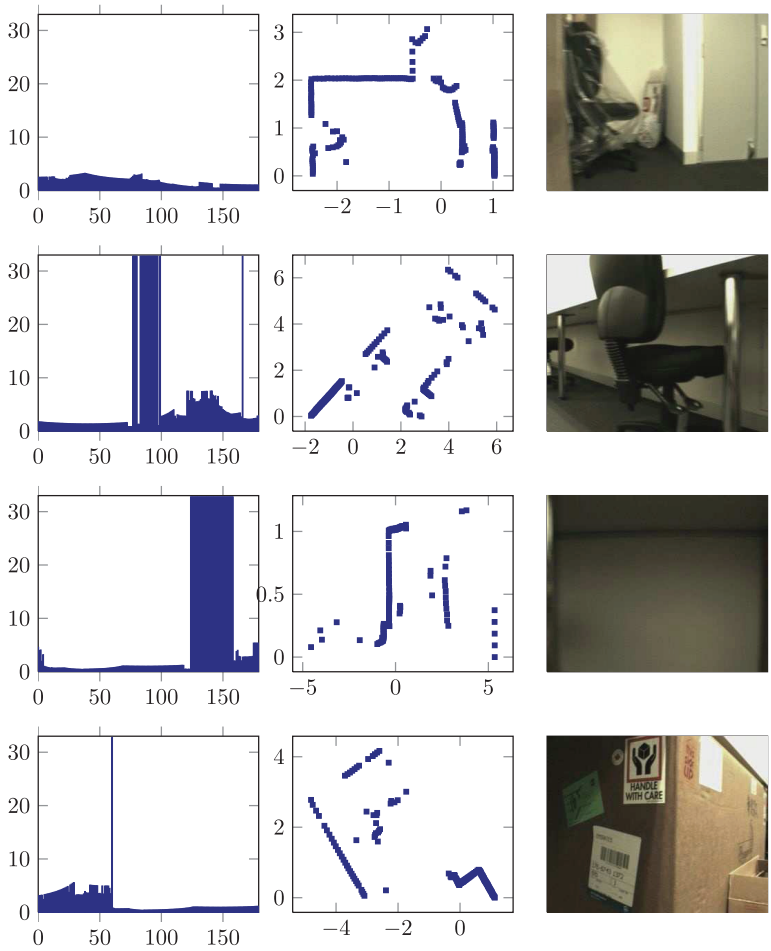


Fig. 11. Some exemplary scans shown in both polar and Cartesian coordinates as well as an image taken by the robot. The first two rows represent cases where no collision with the environment is present. The second two rows both contain collisions. The third row is in collision with a metallic table leg to the left while the fourth row is in collision with the big cardboard box in front of the robot.

In an initial experiment the robot moved for 15 min in an environment and collided with it while gathering images. During this time the model was continuously updated. One would expect that in the beginning all observations are new to the system and thus worth keeping. Over time as more and more of the environment is covered fewer and fewer of the observations should contain new information. This expected behavior is verified by plotting the percentage of observations that were novel to the system over time in Figure 13. We can see that for all three environments in the first few minutes, when the robot is still mostly traversing undiscovered areas, the majority of the observations contain novel knowledge for the system. Then as time progresses most observations can already be explained by the learned model and are thus uninteresting. Every now and then a larger amount of observations contains information which can be explained by the robot actually looking at something it has not seen before or changes in the lighting conditions producing observations that are different enough for the model to not be able to explain them.

A visual representation of the model this system learns in an indoor environment is shown in Figure 12. The grid map is not part of the model; it is just shown for visualization purposes. The patches show the appearance the robot perceives at those locations. A green border indicates a non-obstacle while a red border indicates an obstacle. We can easily see how the carpet floor is recognized as not being an obstacle while other things such as walls, doors and boxes are picked up and correctly labeled as obstacles.

Exemplars found by the clustering in both indoor and outdoor scenarios are shown in Figure 14. As can be seen the clusters found can be easily distinguished from each other and represent the different types of object present in the environment, such as floors, pavement, walls and predominant obstacles. Ideally clusters should be distinct from each other, that is, far apart in the feature space. However, clusters should also contain a reasonable amount of data points, in other words, allow for a certain amount of variability within a cluster. The examples of cluster members shown in Figure 15 demonstrate that the clusters obtained exhibit this property. Each row in Figure 15 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

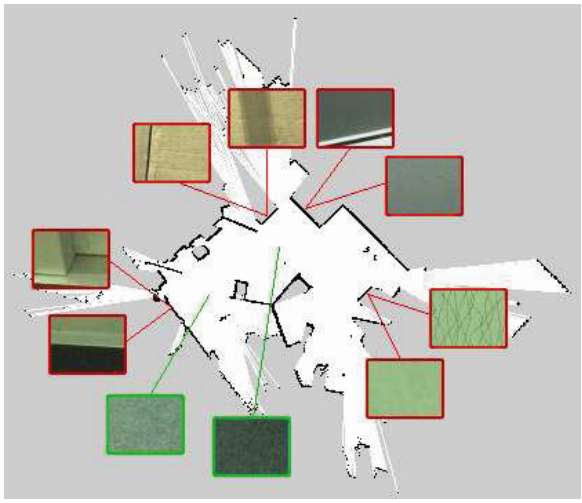


Fig. 12. Map of one of the indoor environments with exemplary image patches referenced to the location they have been perceived at. The colored box around the patches also indicates if they are an obstacle (red) or not (green).

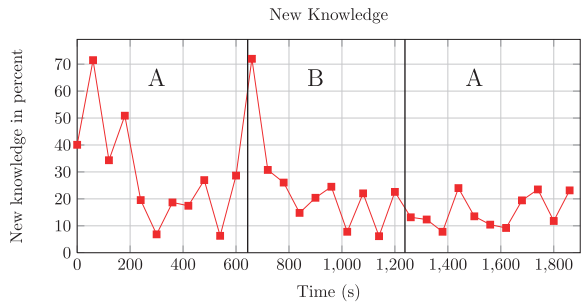


Fig. 17. This plot shows the percentage of novel observations when the method is presented with two types of environment starting with environment A then switching to environment B, and then back to environment A. The black vertical lines indicate the change from one environment into the other. This clearly shows two desirable properties. Upon entering a novel environment most of the observations are novel and need to be represented. Upon switching back into the already-known environment A no such spike in novelty occurs which is exactly what we would expect.

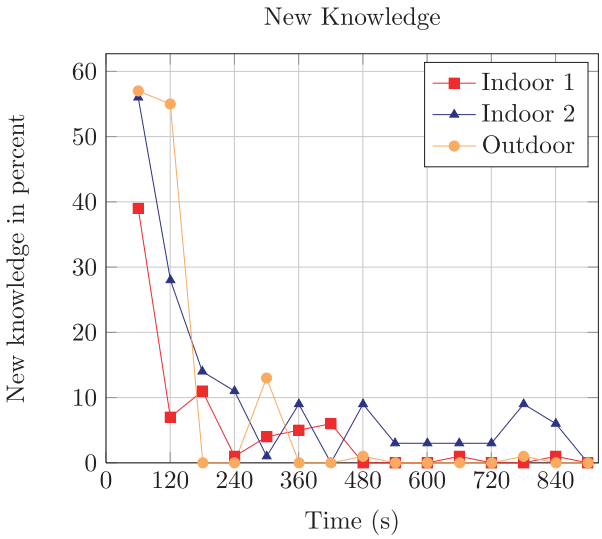


Fig. 13. The plot shows the percentage of all observations made during a 60 s window that add new information to the environment model. It is clear that the majority of observations which lead to new knowledge are observed in the beginning.

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 appearances, allows the overall number of clusters to be kept small and thus more representative of the environment.

5.3.2. Model consistency for long-term autonomy. The previous section shows that we can successfully learn a model of the environment from scratch without human supervision. However, for long-term autonomy we not only need to be able to build an initial model but also to maintain it in the event that the environment changes. To demonstrate this we had the robot move in two visually distinct outdoor environments and move between them. Figure 16 shows exemplary image patches from the areas used in the experiment. The robot starts out in environment A and moves there for a while before moving to environment B, and returns to environment A after a while. The plot of novelty of observations over time for this experiment is shown in Figure 17. The two black vertical bars in the plot indicate the transition between the two environments. At the first transition there is a clear spike in the amount of new knowledge contained in the observations, as we visit a visually novel area. When moving back from area B into area A there is no such spike as the model learned previously in area A can be reused. To simulate an operation over an extended period of time the above experiment is repeated with data gathered several months after the initial visit of area A, and referred to as A'. We perform the same task as in the previous experiment, moving from A to B and then to A'. The novelty of observations for this are shown in Figure 18. We can see a behavior very similar to that in Figure 17. We have spikes in the novelty when first visiting areas A and B but not when entering A'. The novelty values of area A' are not as low as in the previous experiment, however, they still indicate that major parts of the existing model are reused even though the original model is based on data gathered months prior to this. While this experiment is not performed with a robot operating for an extended period of time it still shows that the method can deal with changes that occur over time by reusing an existing model and adding to it. These experiments show how the proposed system can learn valid models from observations and adapt and update the model when changes occur over time without compromising the current model.

Fig. 14. Examples of the exemplars as determined by STRAP. Results from indoor experiments are shown on the left while on the right exemplars obtained in outdoor experiments are shown.

Fig. 15. 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 color).

5.3.3. Obstacle avoidance performance. In order to test the capability of our method to detect and avoid obstacles we placed obstacles in the environment for the robot to detect. The system was allowed to automatically learn a model of the environment 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 recognized and stopped at a distance of $0.94\,\text{m} \pm 0.23\,\text{m}$. This 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.

Fig. 16. Exemplary images from the different areas, from top to bottom: A, A' and B. Areas A and A' differ mainly in illumination and camera setting changes, which affects the color appearance.

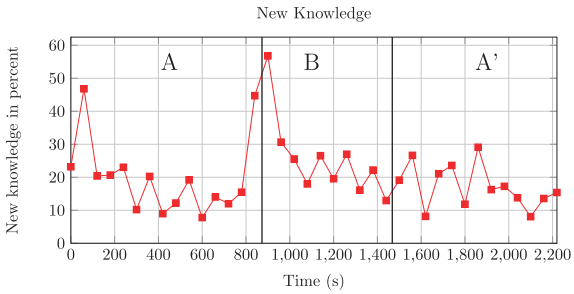


Fig. 18. This plot visualizes the novelty of observations when the data from areas A and A' is recorded several months apart, emulating long-term operation. From the graph we can see that although the data was recorded several months apart the method still is able to reuse most of the existing model when visiting A'.

6. Conclusion

We propose to approach the long-term autonomy problem in robotics by enabling robots to learn and maintain models of their environment without the need for human supervision. To this end, we improve the computational efficiency of affinity propagation, the clustering method used in our methods, by introducing meta-point affinity propagation. In experiments we show how this new method has a similar clustering quality to affinity propagation but at a significantly lower computational cost. We then show how the proposed methods can be used on a real robotic system to learn to predict collisions with the environment from raw laser scanner readings without any information provided by a human expert. Finally, we show how a robot can build a model of the visual appearance of its environment in an unsupervised way. Making use of the bumper of the robot we can additionally self-supervise it to learn which objects identified by the model represent obstacles. This combination allows a robot to build a model of the environment and learn to avoid obstacles without any human help.

Overall the proposed methods give robots the capability to build a model of their environment and adapt to changes in the environment in an entirely unsupervised manner. This is a crucial skill for the long-term autonomy of robots as we cannot assume that the environment remains unchanged over days, weeks or months. It additionally makes the deployment of a robotic system easier and quicker as no human labor is required to build the model of the environment the robot is going to operate in.

Funding

This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

References

Agarwal S and Roth D (2002) Learning a sparse representation for object detection. In: *Proceedings of the European conference on computer vision (ECCV)*.

Blei D, Ng A and Jordan M (2003) Latent Dirichlet allocation. *Journal of Machine Learning Research* 3: 993–1022.

Cao F, Ester M, Qian W, et al. (2006) Density-based clustering over an evolving data stream with noise. In: *SIAM International conference on data mining*.

Ester M, Kriegel H, Sander J, et al. (1996) A density-based algorithm for discovering clusters in large spatial databases with noise. In: *Proceedings of the International conference on knowledge discovery and data mining*.

Fergus R, Perona P and Zisserman A (2003) Object class recognition by unsupervised scale-invariant learning. In: *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*.

Frey B and Dueck D (2007) Clustering by passing messages between data points. *Science* 315: 972–976.

Giguere P, Dudek G, Prahacs C, et al. (2009) Unsupervised learning of terrain appearance for automated coral reef exploration. In: *Proceedings of the Canadian conference on computer and robot vision*.

Happold M, Ollis M and Johnson N (2006) Enhancing supervised terrain classification with predictive unsupervised learning. In: *Proceedings of robotics: Science and systems (RSS)*.

Howard A, Turmon M, Matthies L, et al. (2006) Towards learned traversability for robot navigation: From underfoot to the far field. *Journal of Field Robotics* 23: 1005–1017.

Katz R, Nieto J and Nebot E (2010) Unsupervised classification of dynamic obstacles in urban environments. *Journal of Field Robotics* 27: 450–472.

Kim D, Sun J, Rehg J, et al. (2006) Traversability classification using unsupervised on-line visual learning for outdoor robot navigation. In: *Proceedings of the IEEE International conference on robotics & automation (ICRA)*.

Kschischang F, Frey B and Loeliger HA (2001) Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory* 47: 993–1022.

Maier D, Bennewitz M and Stachniss C (2011) Self-supervised obstacle detection for humanoid navigation using monocular vision and sparse laser data. In: *Proceedings of the IEEE International conference on robotics and automation (ICRA)*.

Meyer F and Beucher S (1990) Morphological segmentation. *Journal of Visual Communication and Image Representation.*

Modayil J and Kuipers B (2004) Bootstrap learning for object discovery. In: *Proceedings of the IEEE/RSJ International conference on intelligent robots and systems (IROS)*.

Ng A, Jordan M and Weiss Y (2001) On spectral clustering: Analysis and an algorithm. In: *Proceedings of advances in neural information processing systems*.

Ojala T, Pietikainen M and Maenpaa T (2002) Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24: 971–987.

Pearl J (1988) *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann.

Rosenberg A and Hirschberg J (2007) V-measure: A conditional entropy-based external cluster evaluation measure. In: *Proceedings of the joint conference on empirical methods in natural language processing and computational natural language learning*.

Ruhnke M, Steder B, Grisetti G, et al. (2009) Unsupervised learning of 3D object models from partial views. In: *IEEE International conference on robotics and automation*.

Stavens D and Thrun S (2006) A self-supervised terrain roughness estimator for off-road autonomous driving. In: *Proceedings of the conference on uncertainty in AI*.

Steinberg D, Pizarro O, Williams S, et al. (2010) Dirichlet process mixture models for autonomous habitat classification. In: *OCEANS 2010 IEEE*.

Ulrich I and Nourbakhsh I (2000) Appearance-based obstacle detection with monocular color vision. In: *Proceedings of the National conference on artificial intelligence*.

Weber M, Welling M and Perona P (2000) Unsupervised learning of models for recognition. In: *Proceedings of the European conference on computer vision (ECCV)*.

Zhang X, Furtlehner C and Sebag M (2008) Data streaming with affinity propagation. In: *Proceedings of the European conference on machine learning and principles and practice of knowledge discovery in databases*.