THE UNIVERSITY OF
SYDNEY

Unsupervised Learning for Long-Term Autonomy

Lionel Ott

A thesis submitted in fulfillment
of the requirements for the degree of
Doctor of Philosophy

THE UNIVERSITY OF
SYDNEY

Faculty of Engineering and Information Technologies
University of Sydney

2015

Declaration

I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person nor material which to a substantial extent has been accepted for the award of any other degree or diploma of the University or other institute of higher learning, except where due acknowledgement has been made in the text.

Lionel Ott

27 August 2014

Abstract

This thesis investigates methods to enable a robot to build and maintain an environment model in an automatic manner. Such capabilities are especially important in long-term autonomy, where robots operate for extended periods of time without human intervention. In such scenarios we can no longer assume that the environment and the models will remain static. Rather changes are expected and the robot needs to adapt to the new, unseen, circumstances automatically.

The approach described in this thesis is based on clustering the robot's sensing information. This provides a compact representation of the data which can be updated as more information becomes available. The work builds on affinity propagation (Frey and Dueck, 2007), a recent clustering method which obtains high quality clusters while only requiring similarities between pairs of points, and importantly, selecting the number of clusters automatically. This is essential for real autonomy as we typically do not know *"a priori"* how many clusters best represent the data.

The contributions of this thesis are three fold. First, a self-supervised method is presented that automatically builds a visual appearance model of the environment with traversability information of objects within the environment. The model allows the detection of obstacles in new observations for safe navigation. Improvements over existing large scale affinity propagation extensions are made with meta-point affinity propagation, reducing computational time and improving robustness to noise. Experiments on indoor and outdoor data demonstrate the quality of the model as well as the ability to recognise obstacles. The usefulness in long-term autonomy is demonstrated on outdoor data captured months apart.

Second, an extension of affinity propagation capable of clustering multiple data sources automatically is described. The method only requires information on a per sensor basis, which is automatically combined into a single clustering solution. This is of particular interest in robotics, as robots generally obtain data from different sensors. Experiments on indoor and outdoor RGB-D data demonstrate the advantage of combining multiple sensor modalities and the quality obtained by the proposed method in tasks such as segmentation and clustering.

Third, joint clustering and anomaly detection is formulated as a generic integer program to which two optimisation methods are proposed. The formulation

automatically selects the number of clusters as well as a user defined number of meaningful outliers while only requiring pairwise similarities as input. The first solution is formulated using the affinity propagation framework while the second uses the Lagrangian duality which provides guarantees on the optimality of the solution. The benefits and properties of the proposed methods, including optimality, scalability and robustness, are shown on synthetic data. Applicability to real data is demonstrated on various real datasets such as hurricane GPS traces, MNIST digits, and New College image dataset.

Acknowledgements

First and foremost I would like to thank my supervisor Fabio Ramos for the opportunity to pursue a PhD with him, as well as his readily available support during the four years. The ability to explore freely while always having the ability to get feedback was immensely helpful.

I would also thank the University of Sydney and NICTA for the scholarship which allowed me to undertake this research.

I would also like to thank all the other students in Fabio's group for interesting chats on various topics and especially the trips to the Field Lab for a dose of coffee. Thanks also to my family, which offered support despite being at the other side of the globe.

Nomenclature

Notation

The following is a listing of the notation used throughout this thesis.

General

X	matrix
x	vector
x^T	transpose of vector x
$\|x\|$	L^2 norm of vector x
$\|x\|$	cardinality of x
N	number of points in a dataset
ℓ	number of outliers
k	number of clusters to select
μ	centroid of a cluster
$\mathcal{N}(\mu, \Sigma)$	normal distribution with mean μ and covariance Σ
$\mathcal{B}^k$	k-th laser scan bin
$\mathcal{B}_i^k$	i-th measurement of the k-th laser scan bin
$\mathcal{S}$	laser scan

Spectral clustering

S	similarity matrix
G	similarity graph
D	degree matrix
W	weight matrix
L	graph Laplacian matrix
U	projected data matrix

Convex optimisation

f, g, q	functions
x	function input
$x^{[t]}$	function input at time t
X	value range of x
$\mathcal{L}$	Lagrangian function
λ	Lagrangian multiplier
f^*, q^*	optimal function values
$u^{[t]}$	step size at time t
$\Delta x^{[t]}$	step direction at x at time t
$\nabla f(x)$	gradient of f at x
$s^{[t]}$	subgradient at time t
$\delta f(x)$	subdifferential of f at x

Features and metrics

$d(i, j)$	distance between point i and j
l, a, b	LAB colour space values
x, y	image pixel coordinates
$\mathcal{H}$	histogram
H	entropy
$\mathcal{I}$	image
p	image pixel
$dist(\mathcal{I}_a, \mathcal{I}_b)$	distance between two images

Affinity propagation

$C(e,a), C(X)$	cost of specific solution
e	exemplar choices
a	assignments of points to exemplars
X	binary assignment matrix
x_{ij}	assignment matrix value in row i column j
I, E, S, Q, P	factor nodes
L	layer in layered affinity propagation
$\beta_{ij}, \eta_{ij}, \rho_{ij}, \alpha_{ij},$ $\tau_{ij}, \omega_{ij}, \chi_{ik}, \lambda_{ik},$ τ_{ik}, ω_{ik}	factor graph message
S	similarity matrix
s_{ij}	similarity value between point i and j
$\mu_{v \to f}$	variable to factor message
$\mu_{f \to v}$	factor to variable message
f	factor in a factor graph
v	variable node in a factor graph
$\mathcal{M}_{long}$	long term model represented by streaming affinity propagation
$\mathcal{M}_{short}$	short term model represented by affinity propagation
$\theta_{\text{min-points}}$	minimum number of points required for a meta-point to be considered during the clustering
$\theta_{similarity}$	radius of a meta-point sphere
$\mathcal{P}$	set of points used for clustering with MPAP
$\mathcal{N}$	set of points considered noise with MPAP

Lagrangian duality

c_j	cost of selecting point j as exemplar
d_{ij}	cost of assigning point i to exemplar j
o	outlier indicator vector
A	constraint matrix
FLO_{LP}	optimal solution value of the linear program
FLO_{LR}	optimal solution value of the Lagrangian relaxation
FLO_{IP}	optimal solution value of the integer program

Abbreviations

AP	Affinity propagation
APOC	Affinity propagation outlier clustering
BIC	Bayesian information criterion
FLO	Facility location with outliers
GPS	Global positioning system
IMU	Inertial measurement unit
IP	Integer program
LAP	Layered affinity propagation
LBP	Local binary pattern
LD	Lagrangian duality
LDA	Latent Dirichlet allocation
LOF	Local outlier factor
LP	Linear program
LR	Lagrangian relaxation
MAP	Maximum a posteriori
MPAP	Meta-point affinity propagation
RANSAC	RANdom SAmple Consensus
ROC	Receiver operating characteristic
STRAP	Streaming affinity propagation
TU	Totally unimodular

Contents

List of Figures

List of Tables

List of Algorithms

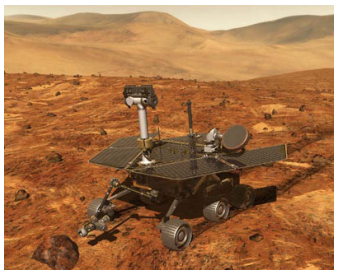
Chapter 1

Introduction

Robotics has evolved drastically over time, from tools used to automate manufacturing to autonomous rovers exploring Mars and self-driving cars. This would not have been possible without the advances and development of sophisticated hardware, from improved robotic designs to sensors such as IMU, cameras and laser scanners, associated with theoretical developments in machine learning. Challenges to the robotics community, such as the ones organised by DARPA, play a significant role in advancing what is currently feasible. Projects such as the Google car would hardly be at the point they are today without those challenges. Robots are also becoming ubiquitous in every day life, from automated vacuum cleaners to small quadrotors. This means that robots are used in areas and ways that were not thought possible only a short time ago, naturally creating new challenges, such as operating robots for extended periods of time with no or minimal supervision by humans. The availability of robots to non experts also highlights the importance of methods that do not rely on highly specialised knowledge to obtain good results. Robotics has always been a field which combined research from many different fields. Machine learning, supervised learning methods in particular, have been used extensively thanks to their ability to build accurate models from labelled data. However, for long-term autonomy, the big question is how can robots operate in unknown, changing environments without requiring human supervision?

1.1 Motivation

With robots being deployed to operate longer, with less human supervision, and by non-experts, methods with the ability to adapt become paramount. In such scenarios an important question is how models built based on training data can adapt to changes in the environment? One solution is to label more data and retrain the model. However, this is not appealing in the context of long-term autonomy due to the required human involvement. Unsupervised learning methods, capable of automatically building and adapting a model, are an interesting solution to these problems. Such methods are not without challenges, as data



(a) Mars Rover

(b) Google Car

(c) Parrot AR Drone

Figure 1.1: Contemporary robots of varying complexity deployed in different scenarios. From Mars rovers (a) (courtesy NASA/JPL-Caltech) to the autonomous Google car (b) (Courtesy of Wikimedia Commons) to recreational drones (c) (http://www.parrot.com/).

needs to be processed continuously and with limited control over the data being gathered. Clustering methods are good candidates as they typically make little to no assumptions about the data, and purely work on distance metrics to group the data into groups. This allows the clusters, or the model, to change as more and different data is observed.

The use of unsupervised methods has other advantages in addition to adapting to changes. There is no need for an initial model to exist and, as such, enables the creation of models entirely from scratch. This is particularly appealing for deployments in previously unvisited or inaccessible areas, such as other planets or disaster zones. Finally, such methods are suitable for use by non experts as they can independently learn a model after being deployed.

In another scenario, assume a robot equipped with a camera and a bump sensor is required to build a model that allows it to safely navigate the environment, i.e. without collisions. A typical supervised approach would label images collected in the environment with obstacle information, and train a classifier based on this data, allowing the robot to identify obstacles in images. However, as soon as the environment changes the model is no longer valid. Using unsupervised methods this issue can be avoided altogether. At the beginning, the model is empty and has to be built incrementally by observing the environment. Clustering images can be used to build a model of the environment, however, the information needed to classify parts of the environment as obstacles or non-obstacles is still missing. This information is added by incorporating the bump sensor in a self-supervised manner. Whenever the robot collides with the environment the bump sensor is triggered and the currently observed image data is labelled as belonging to obstacles. Over time the robot will accumulate this information in the clusters and be able to infer that certain clusters belong to obstacles. As time continues fewer collisions will occur, as the model will be able to classify observations correctly

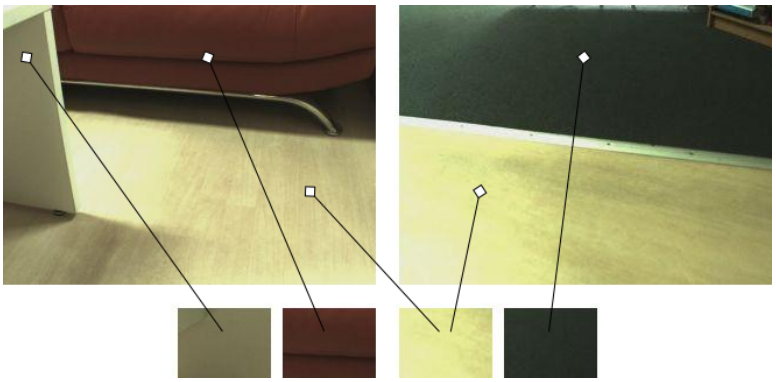


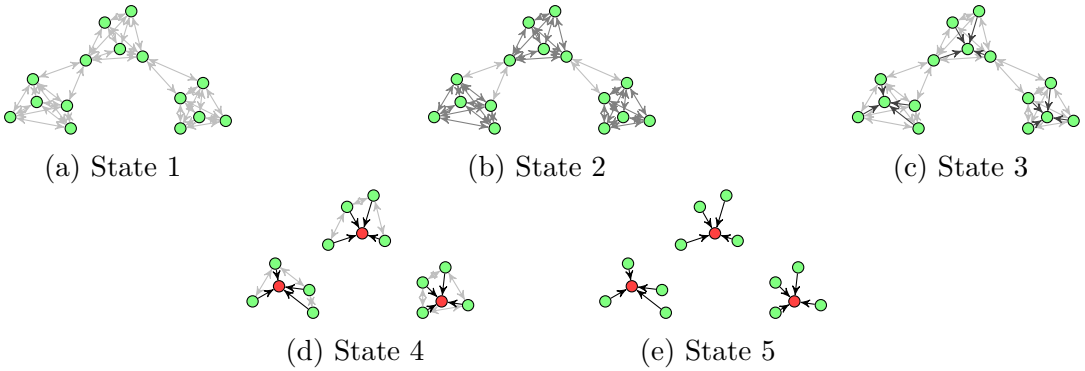
Figure 1.2: Example of the exemplars and their relation to parts of the image.

into obstacles and non-obstacles. Figure 1.2 shows what such a model looks like, the small image patches indicate the clusters, corresponding to different objects in the environment. The two patches on the left represent obstacles while the two on the right correspond to non-obstacles. All of this happens without human supervision and is therefore amenable to adaptation in a changing environment.

1.2 Affinity propagation

Clustering algorithms, on which methods developed in this thesis are based on, group data points into clusters according to how similar or close they are to each other. Clustering methods typically require the user to provide the number of clusters to create beforehand, which is impossible for applications such as the one described above. Affinity propagation (Frey and Dueck, 2007) is a clustering method that does not require this information and works solely based on the distances between pairs of data points.

A short overview of affinity propagation is provided here with more details available in Section 2.2. Affinity propagation requires a square similarity matrix which contains the similarity or distance between all pairs of points as input and produces clusters and exemplars – the most representative point of a cluster – as outputs. Based on the similarity matrix a factor graph is built representing the clustering problem. The goal is to find the *maximum a posteriori* solution to the factor graph which is achieved through message passing, which propagates messages between the nodes in the factor graph. The messages represent two intuitive measures: *availability*, sent from point k to point i, indicates how appropriate it is for a point i to choose k as its exemplar, and *responsibility*, sent from point i to point k, indicates how suitable k is as an exemplar for i. By iteratively updating these two messages across the entire graph a clustering solution emerges over time. The evolution of this process is depicted in Figure 1.3, where circles indicate the different data points, and arrows the availability messages,



(a) State 1 (b) State 2 (c) State 3

(d) State 4 (e) State 5

Figure 1.3: Different states in an exemplary run of affinity propagation. The arrows indicate the availability 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, marked in red, emerge.

with darker ones having higher belief value. Over time the clusters disconnect from each other and the exemplar points emerge.

While affinity propagation allows us to cluster data without providing an explicit number of clusters, it has drawbacks over simpler clustering methods such as k-means. The biggest drawback is runtime, which is quadratic in the number of data points making scaling to large amounts of data, and incorporating new data in a streaming manner, challenging. Additionally, the formulation of affinity propagation makes extensions challenging. Finally, affinity propagation has to cluster all data points which can be problematic in the presence of noise.

1.3 Problem statement

In its most general form the problem addressed in this thesis is: how to enable a robot to learn and maintain a model of the environment autonomously. Building the model should not require any human supervision. The methods should be incremental and run in real-time. Not relying on human supervision allows the robot to be used in a much wider range of applications while incremental operation allows the robot to adapt to changes without throwing away previous efforts. The very general nature of these requirements makes the developed methods usable within a wide range of sensors and data, including cameras, laser scanners, GPS traces, and more. The models built in this way can be augmented with experiences made by the robot in a self-supervised manner, resulting in models that are automatically customised to the robot's sensors and the task to be performed.

1.4 Contributions

This thesis addresses the challenges outlined above and makes contributions to clustering in general with applications to long-term autonomy. In the following we detail each of the contributions.

1.4.1 Self-supervised model learning

Robots operating for extended periods of time require methods that enable them to build and maintain a model of the environment without human supervision. We propose to address this problem using unsupervised methods, namely clustering. As the methods need to operate without human input a clustering method capable of automatically selecting the right number of clusters is needed, affinity propagation in this case. Using this method an appearance model of the environment is built by clustering images observed by the robot. In order to provide information about whether an object is and obstacle or not the bump sensor on the robot is used in a self-supervised manner to annotate clusters with collision information. This allows building an appearance model with obstacle information without human supervision.

While affinity propagation produces good results it has quadratic runtime in the number of data points, which makes it impractical for real-time use in robotics. To address this problem an extension, meta-point affinity propagation, is introduced which reduces the amount of data points considered, and thus runtime, significantly. An additional benefit of this method is that it allows handling outliers, which affinity propagation is not capable of.

1.4.2 Multi-sensor clustering

Clustering algorithms typically require the user to provide distance information for pairs of points. In robotics, however, often times the goal is to cluster multiple data sources jointly which requires building and tuning a function that weights the different data sources against each other, before providing the clustering method with the combined distance values. This process is time consuming and does not generalise easily to different sensor configurations or datasets. The thesis addresses this with an extension to affinity propagation which requires only the distances between pairs of points for each sensor individually. Combing the distances into a single clustering solution is performed by the clustering method in a way that obtains the best possible solution considering all the sources without any further tuning.

This makes processing multi modal data convenient and approachable. Experimental evaluation on RGB-D data shows the benefits of using multiple modalities

and the quality of the results obtained by selecting a clustering solution that is optimal with respect to all data modalities.

1.4.3 Joint clustering with outlier selection

A common phenomena in robotics data is noise, which often times is reduced by filtering the data. However, noise or outliers can also provide us with information. To address this an integer program formulation of clustering with outlier selection is presented. The goal of this is to cluster the data while removing a fixed number of points from the dataset which are the most likely outliers. In doing so the clustering problem becomes easier and more robust while at the same time information about the outliers in the data is preserved.

Two methods to solve the integer program are presented, one based on an extension of affinity propagation to outlier clustering, the other based on the Lagrangian relaxation of the integer program. The Lagrangian method exhibits additional appealing properties such as provable optimality, convergence guarantees, and ease of parallel implementation. Experiments extensively test both methods on both synthetic and real datasets, demonstrating the ability to reveal interesting outliers as well as obtaining near optimal solutions at the fraction of the computational cost required by typical linear program solvers.

1.5 Outline

This chapter showcased how robotics is moving towards longer term deployments with less access to expert human supervision and therefore requires methods capable of unsupervised learning. Affinity propagation is a clustering method that provides a good base to build upon for these goals. This thesis takes advantage of the strengths of affinity propagation while improving its weaknesses and extending it for use in robotics applications.

Chapter 2 provides background knowledge necessary to the understanding of the thesis. Starting with clustering methods in Section 2.1 before describing affinity propagation and its extension to streaming, streaming affinity propagation, in Section 2.2. Section 2.3 provides an overview of convex optimisation and subgradient optimisation. Image data pre-processing methods are detailed in Section 2.4 while Section 2.5 describes the different feature extraction and comparison methods used in the thesis. Finally, Section 2.6 details the evaluation metrics used throughout the thesis.

Chapter 3 presents a system capable of learning a model of the environment and deciding if objects represent an obstacle to the robot without human supervision. Additionally, meta-point affinity propagation is presented as an extension

to affinity propagation which significantly improves the computational performance. The problem is introduced in Section 3.1 and related work is discussed in Section 3.2. The system itself, using affinity propagation and streaming affinity propagation, is discussed in Section 3.3 before the meta-point affinity propagation (MPAP) extension is discussed in Section 3.4. Experiments in Section 3.5 demonstrate the advantages of MPAP, as well as, the capability of the proposed system to learn and maintain a model based on data collected several months apart.

Chapter 4 generalises affinity propagation to allow multiple data sources to be clustered jointly without having to specify a joint cost function. Section 4.1 introduces the problem together with related work in Section 4.2. The graphical model and messages required for the method, layered affinity propagation, are derived in Section 4.3 based on the binary variable model (Givoni and Frey, 2009). Experiments in Section 4.4 demonstrate the capabilities of the method when segmenting and clustering RGB-D data.

In Chapter 5 a linear program formulation of the joint clustering and anomaly detection problem is introduced. The goal is to remove the most likely outliers in the dataset while automatically selecting the number of clusters. The problem is introduced in Section 5.1 and related work is discussed in Section 5.2. Section 5.3 presents the integer program formulation alongside the derivation of the two methods of solving it: (i) a message propagation formulation and (ii) a Lagrangian duality based method. A proof of the optimality of the Lagrangian method as well as a comparison between the two methods follows. In Section 5.4 extensive evaluations on synthetic and real datasets demonstrate the behaviour of the methods and demonstrate the ability to reveal interesting outliers in the data while obtaining good clustering results.

The thesis concludes in Chapter 6 with a summary of the work presented in Section 6.1 and directions for future research in Section 6.2.

Chapter 2

Background

In this chapter we present the theoretical background of methods used throughout this thesis. In Section 2.1 we present clustering algorithms such as k-means and spectral clustering. Section 2.2 gives an introduction to affinity propagation, the clustering method on which many of our contributions are built. This is followed by an overview of convex optimisation and its popular methods in Section 2.3. Section 2.4 introduces common image processing methods used to split an image into smaller patches, something commonly performed prior to feature extraction. Feature extraction and distance functions for use in clustering methods are explained in Section 2.5. Finally, metrics used to compare clustering results are detailed in Section 2.6.

2.1 Clustering

Clustering is an unsupervised learning method that groups data according to the similarity of data points. As such clustering methods require the computation of the similarity or distance between pairs of data points. The number of clusters or groups found by the method are, based on the algorithm, either defined by the user or automatically determined at runtime.

2.1.1 k-means

k-means is a popular clustering algorithm consisting of the following two steps:

1. assign points to cluster centroids;
2. update centroids based on assignments.

These two steps are repeated until the assignments are stable. The algorithm requires the specification of the number of clusters. Algorithm 1 shows the steps performed by the k-means algorithm in more detail. First, the k initial centroids μ_i are selected (line 1 to 3). Next, points p_i are assigned to the centroid closest to them (line 5 to 7) and subsequently the centroids are updated (line 8 to 10).

Algorithm 1: k-means

Input: data points $p_i \in P$, number of clusters k, number of data points N
Output: cluster centroids μ, cluster assignments a

1 **for** $i \in \{1, \ldots, k\}$ **do**
2 $\quad \mu_i \leftarrow p_r \in P$
3 **end**
4 **repeat**
5 $\quad$ **for** $i \in \{1, \ldots, N\}$ **do**
6 $\quad\quad a_i \leftarrow \text{argmin}_j \|p_i - \mu_j\|$
7 $\quad$ **end**
8 $\quad$ **for** $j \in \{1, \ldots, k\}$ **do**
9 $\quad\quad \mu_j \leftarrow (\sum_i p_i \mathbf{I}(i,j)) \,/\, (\sum_i \mathbf{I}(i,j))$
10 $\quad$ **end**
11 **until** *until convergence*
12 **return** μ, a

This is repeated until a convergence criteria, such as no assignment changes, is fulfilled. The $\mathbf{I}$ function in line 9 is an indicator function defined as follows:

$$\mathbf{I}(i,j) = \begin{cases} 1 & \text{if } a_i = j \\ 0 & \text{otherwise} \end{cases}. \tag{2.1}$$

The result of the algorithm are k centroids with the points assigned to the corresponding clusters.

The advantages of k-means are that it is a very simple and highly efficient algorithm with a runtime of $O(N)$, i.e. linear in the number of data points N. Since the assignment of points to centroids is independent of each other this step can also be parallelised easily. The drawbacks of the method are its reliance on the user provided value for k as well as the need for good initial centroid selection. A lot of work has been done in order to address these issues, such as using the Bayesian information criterion (BIC) to determine the number of clusters (Pelleg and Moore, 2000) or smart ways of selecting the initial centroids (Arthur and Vassilvitskii, 2007).

2.1.2 Spectral clustering

Spectral clustering refers to a group of algorithms operating on similar principles (see (von Luxburg, 2007) for a detailed overview). Here we describe a common version of the algorithm. Spectral clustering requires a similarity matrix which stores the pairwise similarities between pairs of points $S_{ij} = s(i,j)$. Based on this a similarity graph G is built by connecting points based on a criterion such

Algorithm 2: Spectral clustering

Input: similarity matrix S, number of clusters k
Output: clustering centroids μ, cluster assignments a

$G \leftarrow$ `BuildSimilarityGraph`$(S, condition)$
for $i, j \in \{1, \ldots, N\}$ **do**
  if $G_{ij} = 1$ **then**
    $W_{ij} \leftarrow S_{ij}$
  end
end
for $i \in \{1, \ldots, N\}$ **do**
  $D_{ii} \leftarrow \sum_j W_{ij}$
end
$L \leftarrow D - W$
for $i \in \{1, \ldots, k\}$ **do**
  $U_i \leftarrow$ `EigenVector(`L, i`)`
end
$\mu, a \leftarrow$ k-`means(`U, k`)`
return μ, a

as ϵ-neighbourhood, k-nearest neighbours or full connectivity. The corresponding weight matrix W is computed as $W_{ij} = S_{ij}, \quad \forall i, j : G_{ij} = 1$. Finally, the degree matrix D of this graph is derived as follows:

$$D_{ii} = \sum_j W_{ij}. \tag{2.2}$$

Based on the degree and weight matrices the graph Laplacian L is computed as:

$$L = D - W. \tag{2.3}$$

From the graph Laplacian L, we compute the k eigenvectors corresponding to the k smallest eigenvalues. These eigenvectors are then assembled into the new data matrix U as follows:

$$U = [u_1, \ldots, u_k] \in \mathbb{R}^{n \times k}, \tag{2.4}$$

i.e. the eigenvectors make up the columns of U. Finally, this new data matrix U is clustered using k-means. Algorithm 2 shows the typical steps performed during spectral clustering as pseudo code. The selection of the number k, used to select the number of eigenvectors and number of clusters is still topic of ongoing research. Common solutions such as, analysing the eigenvalues or eigenvectors has been shown to work well (Zelnik-Manor and Perona, 2004).

2.2 Affinity propagation

In this section we describe the affinity propagation method proposed by Frey and Dueck (2007) based on the binary variable model presented by Givoni et al. (2011). Affinity propagation is an exemplar based clustering method. This means that for each cluster a representative point, the exemplar, is selected. Additionally, affinity propagation also selects the number of clusters based on the pairwise similarities between points and the cost of cluster creation. The pairwise similarities and cluster creation cost are the only parameters the user needs to provide.

We start by describing the approach from an optimisation point of view in Section 2.2.1. In Section 2.2.2, we described an algorithm for solving the problem using belief propagation (Pearl, 1988).

2.2.1 Optimisation view

Our goal is to find the set of exemplars e and assignments a of points to these exemplars which maximises the cost C of the solution, i.e.:

$$C(e, a) = \sum_{i=1}^{N} \text{sim}(i, a_i) + \sum_{j=1}^{k} \text{cost}(e_j), \tag{2.5}$$

where $\text{sim}(i, a_i)$ is the similarity between point i and the exemplar point a_i, $\text{cost}(e_j)$ is the cost of creating a cluster with point j as the exemplar, and k is the number of clusters selected by the algorithm. Note that similarity and cost values need to be always negative, i.e. $\text{sim}_{ij}, \text{cost}_j < 0$. Points that are very similar to each other have a similarity value close to 0. For this reason our goal is to maximise the cost function as this results in a solution which is as close to 0 as possible.

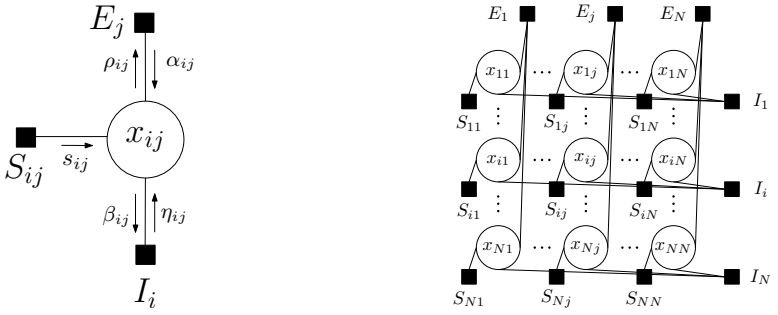
We can simplify Eq. (2.5) by representing the assignments a and exemplars e by a binary assignment matrix X, where $x_{ij} = 1$ indicates that point i is assigned to exemplar j and $x_{jj} = 1$ denotes point j as an exemplar. Additionally we combine the $\text{sim}(i, j)$ and $\text{cost}(j)$ functions into a single matrix as follows:

$$S_{ij} = \begin{cases} \text{cost}(j) & \text{if } i = j \\ \text{sim}(i, j) & \text{otherwise} \end{cases}. \tag{2.6}$$

With these changes the cost function takes the form:

$$C(X) = \sum_{i=1}^{N} \sum_{j=1}^{N} S_{ij} x_{ij} \tag{2.7}$$

In order to obtain valid clustering results we impose the following constraints:



(a) AP Messages (b) AP Graphical Model

Figure 2.1: Visualisation of affinity propagation (a) messages and (b) complete factor graph. The factor graph shows how each row in the assignment matrix X is connected to a single I factor and similarly each column is associated with a single E factor node.

1. an exemplar must choose itself as its own exemplar;
2. points can only be assigned to valid exemplars;
3. every point can only be assigned to a single exemplar.

This results in the following optimisation problem:

$$\text{maximise} \quad \sum_{i=1}^{N}\sum_{j=1}^{N} S_{ij}x_{ij} \tag{2.8a}$$

$$\text{subject to} \quad \sum_{j} x_{ij} = 1 \quad \forall i \tag{2.8b}$$

$$x_{ij} \leq x_{jj} \quad \forall i, j \tag{2.8c}$$

$$x_{ij} \in \{0, 1\} \tag{2.8d}$$

In the next section we describe how Eq. (2.8) can be reformulated such that loopy belief propagation (Kschischang et al., 2001) can be used to solve it.

2.2.2 Graphical model view

Affinity propagation maximises the energy of Eq. (2.8) by representing the problem as a factor graph, shown in Figure 2.1b. The nodes in the graph are the variable assignments x_{ij} while the factors encode the different constraints enforcing a valid solution. The maximum a posteriori (MAP) solution is then computed using loopy belief propagation. The only input required is the similarity matrix S which contains the similarity between pairs of data points and the cost of selecting a point as an exemplar on the diagonal.

The constraints imposed on a valid solution in Section 2.2.1 are formulated as follows:

1. 1-of-N Constraint (I_i). Each data point has to choose exactly one exemplar.

2. Exemplar Consistency Constraint (E_j). For point i to select point j as its exemplar, point j must declare itself an exemplar.

Mathematically, these constraints are formulated as follows:

$$S_{ij}(x_{ij}) = \begin{cases} S_{ij} & \text{if } x_{ij} = 1 \\ 0 & \text{otherwise} \end{cases} \tag{2.9}$$

$$I_i(x_{i:}) = \begin{cases} 0 & \text{if } \sum_j x_{ij} = 1 \\ -\inf & \text{otherwise} \end{cases} \tag{2.10}$$

$$E_j(x_{:j}) = \begin{cases} 0 & \text{if } x_{jj} = \max_i x_{ij} \\ -\inf & \text{otherwise} \end{cases} \tag{2.11}$$

where $x_{i:} = x_{i1}, \ldots, x_{iN}$ and $x_{:j} = x_{1j}, \ldots, x_{Nj}$.

Combining these constraints with the user provided similarity values S_{ij}, we obtain the following energy function to be maximised:

$$\max_X \sum_{i=1}^{N} \sum_{j=1}^{N} S_{ij} x_{ij} + \sum_i I_i(x_{i:}) + \sum_j E_j(x_{:j}). \tag{2.12}$$

In order to optimise this energy function with the max-sum algorithm we propagate messages through the factor graph. In their most general form, these messages are defined as follows:

$$\mu_{v \to f}(x_v) = \sum_{f^* \in ne(v) \setminus f} \mu_{f^* \to v}(x_v), \tag{2.13}$$

$$\mu_{f \to v}(x_v) = \max_{x_1, \ldots, x_M} \Big[f(x_v, x_1, \ldots, x_M) + \sum_{v^* \in ne(f) \setminus v} \mu_{v^* \to f}(x_{v^*}) \Big], \tag{2.14}$$

where f is a factor, or a function over a subset of variables, $\mu_{v \to f}(x)$ is the message sent from node v to factor f, $\mu_{f \to v}(x_v)$ is the message from factor f sent to node v, $ne()$ is the set of neighbours of the given factor or node, and x_v is the value of node v.

In Figure 2.1a it can be seen that each node x_{ij} is connected to three factors S_{ij}, I_i and E_j. The messages ρ_{ij} and β_{ij} are sent from nodes to factors and thus are derived using Eq. (2.13). The other three messages s_{ij}, α_{ij} and η_{ij} are sent from factor to node and need to be derived using Eq. (2.14).

In order to obtain an efficient algorithm we employ several strategies to simplify the computations needed to obtain the solution. Since we are using binary variables and absolute scale is not important it is sufficient to compute the dif-

ference between the two possible variable settings, i.e. $\alpha_{ij} = \alpha_{ij}(1) - \alpha_{ij}(0)$. Furthermore, due to the construction of the two constraints in Equations (2.10) and (2.11) only certain assignments are possible which is exploited here to simplify the update equations. Combining these two ideas we arrive at the following update equations:

$$s_{ij} = S_{ij} \tag{2.15}$$

$$\beta_{ij} = s_{ij} + \alpha_{ij} \tag{2.16}$$

$$\eta_{ij} = -\max_{k \neq j} \beta_{ik} \tag{2.17}$$

$$\rho_{ij} = s_{ij} + \eta_{ij} \tag{2.18}$$

$$\alpha_{ij} = \begin{cases} \sum_{k \neq j} \max(0, \rho_{kj}) & i = j \\ \min\left[0, \rho_{jj} + \sum_{k \notin \{i,j\}} \max(0, \rho_{kj})\right] & i \neq j. \end{cases} \tag{2.19}$$

This can be further simplified by expressing ρ_{ij} as follows:

$$\rho_{ij} = s_{ij} + \eta_{ij} = s_{ij} - \max_{k \neq j} \beta_{ik} = s_{ij} - \max_{k \neq j}(s_{ik} + \alpha_{ik}) \tag{2.20}$$

Thus we have recovered the availability (α) and responsibility (ρ) messages of the original affinity propagation formulation (Frey and Dueck, 2007). The two messages that need to be computed iteratively are therefore α and ρ.

The pseudo code for Algorithm 3 shows the steps performed by affinity propagation. The only input is the similarity matrix S which contains the point to point distances on the off-diagonal as well as the cost of declaring a point an exemplar on the diagonal. The algorithm starts by initialising α and ρ messages to 0 and then repeatedly updates ρ and α until convergence is achieved. Convergence is typically achieved when the energy of the solution is stable over a number of iterations. Finding the MAP solution is performed by first extracting the exemplars and then assigning the remaining points to them. Exemplars are the points for which $\alpha_{jj} + \rho_{jj} > 0$, while the assignment of points to exemplars is achieved by assigning point i to the exemplar e which satisfies $\text{argmax}_e(\alpha_{ie} + \rho_{ie})$.

An example of the evolution of the belief represented by the algorithm about the MAP solution is shown in Figure 2.2. Circles represent the nodes while the darkness of the lines indicate the strength of the association of points to each other. Over time the links between nearby points strengthen identifying the exemplars, shown as red circles.

Algorithm 3: Affinity propagation

Input: Similarity matrix S
Output: Exemplars e and assignments a

foreach $i, j \in \{1, \ldots, N\}$ **do**
  $\alpha_{ij}, \rho_{ij} \leftarrow 0$
end
repeat
  foreach $i, j \in \{1, \ldots, N\}$ **do**
    $\rho_{ij} \leftarrow s_{ij} - \max_{k \neq j}(s_{ik} + \alpha_{ik})$
  end
  foreach $i, j \in \{1, \ldots, N\}$ **do**
    $$\alpha_{ij} \leftarrow \begin{cases} \sum_{k \neq j} \max(0, \rho_{kj}) & i = j \\ \min\left[0, \rho_{jj} + \sum_{k \notin \{i,j\}} \max(0, \rho_{kj})\right] & i \neq j. \end{cases}$$
  end
until *convergence*
$e \leftarrow$ points for which $(\alpha_{jj} + \rho_{jj}) > 0$ holds
$a_i \leftarrow$ assign point i to exemplar e satisfying $\operatorname{argmax}_e(\alpha_{ie} + \rho_{ie})$
return e, a

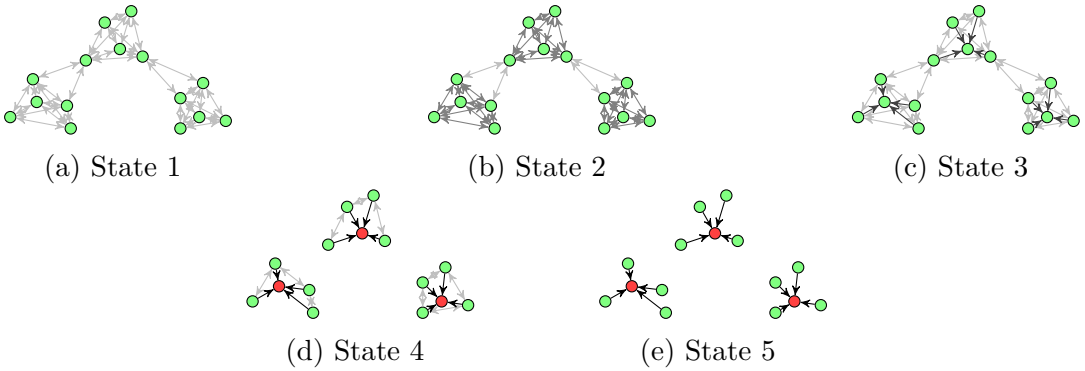
2.2.3 Streaming affinity propagation

While affinity propagation can compute solutions involving a few thousands data points in under a minute, it is not fast enough for use in real-time robotics applications with a large number of observations. However, there are methods extending affinity propagation to handle data streams in real time, such as streaming affinity propagation by Zhang et al. (2008). The naïve approach to use affinity propagation for data streaming would be to recompute the clustering with 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 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 in the cluster, Σ_i



(a) State 1 (b) State 2 (c) State 3

(d) State 4 (e) State 5

Figure 2.2: Different states in an exemplary run of affinity propagation. The arrows indicate the availability 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, marked in red, emerge.

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. 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 and 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 performed, the number of data points involved is small.

2.3 Convex optimisation

Affinity propagation is an optimisation problem, which is solved using message passing, however, classic optimisation is another way to solve the problem. Convex optimisation in particular is appealing due to its properties and is used in Chapter 5 to perform clustering with outlier selection.

Convex optimisation (Boyd and Vandenberghe, 2009) is a field of optimisation concerned with the optimisation of convex functions. A function $f : \mathbb{R}^n \mapsto \mathbb{R}$ is convex if its domain is a convex set and if for all $x, y \in \mathbf{dom} f$ with $0 \leq \theta \leq 1$ the following holds:

$$f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta)f(y). \tag{2.21}$$

A graphical interpretation of this expression is shown in Figure 2.3. Convex functions are desirable as they guarantee that a local optimum is also the global optimum, and as such, can be optimised using simple gradient based approaches.

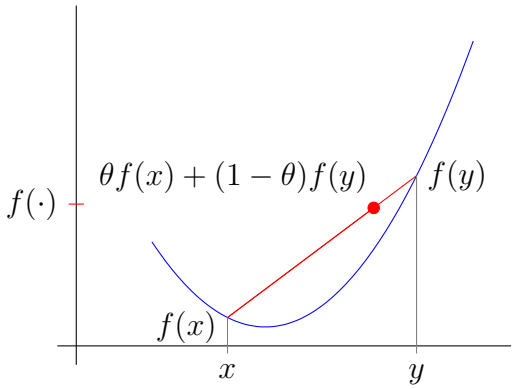


Figure 2.3: Visualisation of the convexity definition of Eq. (2.21).

A general convex optimisation problem has the following form:

$$\begin{aligned} \text{minimise} \quad & f(x) && \text{(2.22a)} \\ \text{subject to} \quad & g_i(x) \leq 0, i = 1, \ldots, m && \text{(2.22b)} \\ & x \in X && \text{(2.22c)} \end{aligned}$$

where f is a convex cost function, g_i the set of convex constraint functions and $X \subset \mathbb{R}^n$ a non-empty closed convex set.

2.3.1 Lagrangian relaxation

The formulation of Eq. (2.22) is called the primal problem. Introducing Lagrange multipliers $\lambda_i \geq 0$ for the constraints g_i, the Lagrangian can be written in the following manner:

$$\mathcal{L}(x, \lambda) = f(x) + \sum_{i=1}^{m} \lambda_i g_i(x). \tag{2.23}$$

This forms a weighted sum of the objective and constraint functions with associated Lagrange multipliers. The optimisation problem formed by the Lagrangian relaxation is called the dual problem and is of the form:

$$\begin{aligned} \text{maximise} \quad & q(\lambda) && \text{(2.24a)} \\ \text{subject to} \quad & \lambda \geq 0 && \text{(2.24b)} \\ & \lambda \in \mathbb{R}^m, && \text{(2.24c)} \end{aligned}$$

where $q(\lambda)$ is the dual function defined by:

$$q(\lambda) = \inf_{x \in X} \left\{ f(x) + \sum_{i=1}^{m} \lambda_i g_i(x) \right\}. \tag{2.25}$$

The optimal value of the primal problem is denoted f^* while the optimal solution to the dual problem is denoted as q^*. This formulation has the following useful property.

Theorem 1. *The dual function $q(\lambda)$ is a lower bound to the primal solution for $\lambda \geq 0$, i.e.:*

$$q(\lambda) \leq f(x^*) \tag{2.26}$$

Proof. Consider a feasible solution $\widetilde{x}$:

$$\begin{aligned} q(\lambda) &= \inf_{x \in X} \mathcal{L}(x, \lambda) \leq \mathcal{L}(\widetilde{x}, \lambda) \\ &= f(\widetilde{x}) + \sum_{i=1}^{m} \lambda_i g_i(\widetilde{x}) \\ &\leq f(\widetilde{x}) \end{aligned} \tag{2.27}$$

since $g_i(\widetilde{x}) \leq 0$ and $\lambda_i > 0$. Therefore, $q(\lambda) \leq \mathcal{L}(\widetilde{x}, \lambda) \leq f(\widetilde{x})$ for all feasible $\widetilde{x}$ and thus $q(\lambda) \leq f(x^*) = f^*$. □

From Theorem 1 we obtain $q^* \leq f^*$. If the inequality is strict, i.e. $q^* < f^*$ we have "weak duality", which gives rise to the duality gap $f^* - q^*$. This gap indicates how far apart the two solutions are. The second and more interesting case, where $q^* = f^*$, is called strong duality and implies that the solution of the dual is equivalent to that of the primal. Strong duality is interesting, as it allows us to solve the primal problem to the optimality by solving another, potentially easier problem. In order for strong duality to hold the primal problem needs to satisfy the so called constraint qualifications.

All of the above becomes more useful when combined with the fact that the Lagrangian dual function $q(\lambda)$ is always a concave optimisation problem, as its objective function is the infimum of a family of affine functions in λ. This means that we can apply standard gradient based optimisation methods to solve it. In the following we will introduce gradient descent and its adaptation to non differentiable functions, subgradient descent (Bertsekas, 1999).

2.3.2 Gradient descent method

One of the simplest methods for convex optimisation is the gradient descent method. Given a convex function $f(x)$ the method generates a sequence $x^{[t]}, t = 1, \ldots$ with

$$x^{[t+1]} = x^{[t]} + u^{[t]} \Delta x^{[t]}, \tag{2.28}$$

which minimises f. The update involves the "step size" $u^{[t]} > 0$ and "step direction" $\Delta x^{[t]}$. The "step direction" is the gradient of f at the current location, i.e.

Algorithm 4: Gradient descent

Input: function to optimise f, starting point x, stopping threshold η
Output: Optimal value of x

repeat
  $\Delta x \leftarrow -\nabla f(x)$
  Select step size $u > 0$
  $x \leftarrow x + u\Delta x$
until $\|\nabla f(x)\| \leq \eta$
return x

$\Delta x^{[t]} = -\nabla f(x)$. The "step size" can be found using a variety of methods such as exact line search, backtracking line search, fixed step size or decreasing step size. Algorithm 4 shows the steps performed by gradient descent.

2.3.3 Subgradient method

As mentioned before the Lagrangian dual problem is a convex optimisation problem, even if the primal problem is non convex. The optimisation problem though is piecewise linear and convex and therefore non differentiable, which is a requirement for gradient descent based methods. However, we can sidestep this problem by using subgradient based methods.

The vector s is a subgradient of the convex function $f : \mathbb{R}^n \mapsto \mathbb{R}$ at x^* if:

$$f(x) \geq f(x^*) + s^T(x - x^*), \tag{2.29}$$

for all $x \in \mathbb{R}^n$. The subdifferential $\partial f(x^*)$ is the set of all subgradients of f at x^*. Using subgradients we can optimise functions using gradient based methods even if they are not differentiable. Algorithm 5 shows the general steps performed during subgradient optimisation of the Lagrangian dual function. The Lagrange multipliers λ are initialised to random values before repeatedly (i) solving the dual function, (ii) selecting one subgradient at the location obtained as the solution of (i) and then (iii) updating the Lagrange multipliers. This is repeated until the optimal solution is found, which is the case when the subdifferential contains the zero vector.

The method used to select the step size $u^{[t]}$ can have a significant impact on convergence and optimality properties. When the step size fulfils the following constraint:

$$\sum_{t=1}^{\infty} u^{[t]} = \infty \qquad \text{and} \qquad \lim_{t \to \infty} u^{[t]} = 0, \tag{2.30}$$

the algorithm is guaranteed to converge to the optimal solution. A simple step

Algorithm 5: Lagrangian dual function optimisation

Input: Optimisation problem $q(\lambda)$
Output: Optimal solution $\lambda^{[t-1]}$

$\lambda^{[0]} \leftarrow$ random vector $\in \mathbb{R}^n_+$
$t \leftarrow 0$
repeat
  $x^* \leftarrow \inf_{x \in X}\{f(x) + \sum_i \lambda_i g_i(x)\}$
  Pick a subgradient $s^{[t]}$ at $\mathcal{L}(\lambda, x^*)$
  $\lambda^{[t+1]} \leftarrow \max(\lambda^{[t]} - u^{[t]} s^{[t]}, 0), \quad u^{[t]} > 0$
  $t \leftarrow t + 1$
until $0 \in \partial q(\lambda^{[t]})$
return $\lambda^{[t-1]}$

size function that fulfils this constraint is the following,

$$u^{[t]} = u^{[0]}\, \alpha^t \quad \alpha \in (0, 1). \tag{2.31}$$

2.4 Image pre-processing

In this thesis we are often interested in modelling objects in scenes captured by images. As such it is desirable to break the image down into smaller patches to reduce the complexity contained in patch compared to the full image. There are many different approaches to this problem with different trade-offs in runtime and accuracy. In the following we present the two methods used in this work, equal subdivision and super pixel methods.

The visual difference of the patches these methods produce is shown in Figure 2.4 where the top image is produced using equal subdivision and the bottom image with SLIC.

2.4.1 Equal subdivision

This is the simplest method to split an image into smaller patches. The original images is subdivided into a user defined number of equal sized rectangles covering the entire image. This method entirely ignores the contents of the image and, as such is inaccurate at finding borders of objects in a scene, however, it is extremely fast to compute.

2.4.2 Super pixel

Another popular approach is the use of super pixels. A super pixel is a collection of connected pixels in the image that are similar to each other. A super pixel

method segments the entire image into a collection of super pixels attempting to split the image along object borders. Commonly used methods include normalized cuts (Shi and Malik, 2000), watershed based methods (Vincent and Soille, 1991) and the recently proposed SLIC method (Achanta et al., 2012). In this work we use SLIC, as it has better runtime performance then alternative methods and produces good super pixels.

In the following we give a short overview of the SLIC algorithm. The basic idea of the algorithm is to run k-means on the image. Unfortunately, running standard k-means would be too costly so a different strategy is adopted. The assignment step of k-means is restricted in size around each centroid. The algorithm requires two parameters, the number of super pixels k to be generated as well as a compactness parameter m. The algorithm starts by seeding the initial k centroids in a regular grid over the image. Next, the algorithm performs several rounds of k-means in a five dimensional space. The search space consists of the three LAB colour channels of the image as well as the x and y pixel coordinates. A special distance function is used, which normalises the colour and coordinate distance of pixels against each other, and has the following form:

$$d_c(i,j) = \sqrt{(l_j - l_i)^2 + (a_j - a_i)^2 + (b_j - b_i)^2} \tag{2.32}$$

$$d_s(i,j) = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2} \tag{2.33}$$

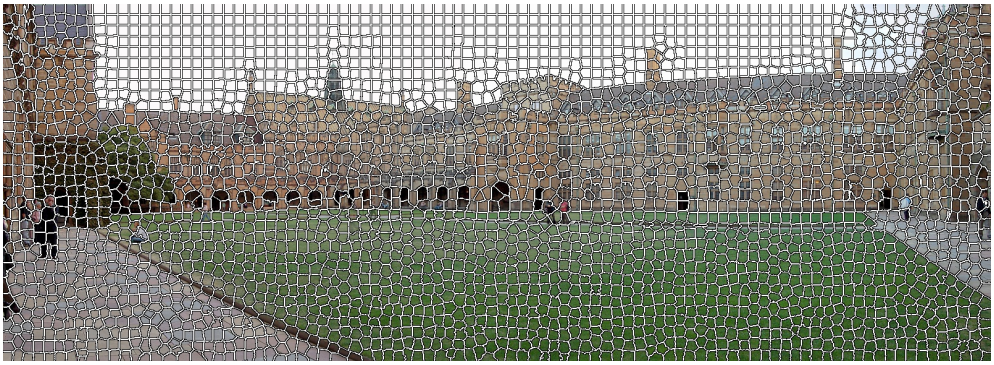
$$D(i,j) = \sqrt{\left(\frac{d_c}{N_c}\right)^2 + \left(\frac{d_s}{N_s}\right)^2}, \tag{2.34}$$

where d_c is the colour distance, d_s the spatial distance, N_c the colour normaliser and N_s the spatial normaliser. The spatial normaliser N_s can be defined as the step size S, the initial distance between neighbouring centroids. The colour normaliser N_c, however, is harder and is often left as a user defined compactness parameter m. Using the normalisers $N_S = S$ and $N_c = m$ the distance function can be rewritten as:

$$D(i,j) = \sqrt{d_c^2 + \left(\frac{d_s}{S}\right)^2 m^2}. \tag{2.35}$$

Tuning m allows to change the behaviour of the super pixels to either be more compact, high value of m, or adhere more to image boundaries, low value of m. The pseudo-code in Algorithm 6 shows the steps involved in the algorithm.

(a) Rectangle subdivision



(b) SLIC

Figure 2.4: Examples of the two presented methods applied to an image of the University of Sydney's Quadrangle. The white border indicates the individual patches.

2.5 Feature extraction and comparison

Typically raw sensor readings need to be processed before being used by other algorithms. This is done for several reasons. First, raw data can be too high dimensional to be used directly especially with images. Second, raw data may not be invariant to certain types of transformations, such as orientation or illumination which makes observations harder to relate to each other. Finally, raw data can be susceptible to noise which a pre-processing step can reduce.

2.5.1 Camera based features

Cameras are a very common passive sensor that provide colour and texture information about a scene and typically have high resolution and information density. Extracting meaningful features from images is a topic of constant research and has evolved dramatically over the years. Examples include simple statistical features, such as the ones we will present here, finely tuned descriptors such as SIFT (Lowe, 1999), and recent advances based on deep belief networks (Lee et al., 2009) and convolutional neural networks (Krizhevsky et al., 2012).

There is always a trade-off between the computational complexity of a feature

Algorithm 6: SLIC

Input: Step size S,
Output: Super pixel labels label

Initialise cluster centres $\mu_k \leftarrow [l_k, a_l, b_k, x_k, y_l]$ using step size S
foreach *pixel* i **do**
    $\text{label}_i \leftarrow -1$
    $\text{dist}_i \leftarrow \infty$
end
repeat
    foreach *Cluster centre* μ_k **do**
        foreach *pixel* i *in a* $2S \times 2S$ *region around* μ_k **do**
            if $distance(i, \mu_k) < \text{dist}_i$ **then**
                $\text{dist}_i \leftarrow distance(i, \mu_k)$
                $\text{label}_i \leftarrow k$
            end
        end
    end
    Update cluster centres
until *iteration limit reached*
Fix connectivity and merge small super pixels
return label

extractor and the quality of the results obtained in a particular application. In the following we will describe very simple histogram based features, as they are very fast to compute and typically offer a reasonable description for our goals.

Colour histograms

Colour histograms are a very simple way to concisely represent the intensity distribution of an image. The intensity value range is discretised into several bins of equal width and the value of each bin is based on the number of pixels that fall into it, i.e.:

$$\mathcal{H}_i = \sum_{p \in \mathcal{I}} \delta(p), \tag{2.36}$$

where $\mathcal{H}_i$ is the i-th bin, $\mathcal{I}$ the image and $\delta(p)$ the Dirac function that returns 1 if p is within the interval of $\mathcal{H}_i$, and 0 otherwise.

Such a representation is good at capturing information on a global scope but loses all spatial information contained in the image. When considering colour images represented by multiple channels, such as RGB, HSV or LAB, different schemes can be used to represent them via histograms. One possibility is to create one histogram per channel and treat them separately. Another one is to build two or three dimensional histograms where each dimension represents a different channel. The choice of which representation is the best often depends on the size

of the image. This is due to the fact that in order to create a meaningful two or three dimensional histogram, a large number of data points, i.e. image pixels, need to be available.

Local binary patterns

As mentioned previously, colour histograms discard the spatial information contained in an image. As such the histograms for two images with identical amount of black and white but one organised as a chequerboard and the other one as two halves of solid colours would result in the same histogram. An attempt to remedy this problem are features which extract texture information from an image. Texture information is typically obtained by evaluating the gradients in a local neighbourhood of a pixel. Aggregating this gradient information into histograms allows us to capture the overall texture information of a single image.

Local binary patterns (Ojala et al., 2002) is a feature that builds such a texture representation. The feature is built by considering P neighbouring points which are equally spaced on a circle of radius R around a central pixel. Based on this the texture extractor is defined as:

$$T = t(g_c, g_0, \ldots, g_{P-1}), \tag{2.37}$$

where g_c is the grey scale value of the centre pixel and $g_0, \ldots, g_{P-1}$ are the values of the neighbouring P pixels.

In order to obtain a feature invariant to global changes in grey scale values and rotation, the following steps are performed. First, the difference between the centre pixel and the neighbours is used, i.e.:

$$T = t(g_c, (g_0 - g_c), \ldots, (g_{P-1} - g_c)). \tag{2.38}$$

This is then simplified via the assumption that $g_i - g_c$ is independent of g_c, yielding:

$$T = t(g_c)t((g_0 - g_c), \ldots, (g_{P-1} - g_c)). \tag{2.39}$$

In practice this assumption is not warranted, however, it provides invariance to shifts in grey scale levels and allows us to remove g_c from the texture extractor. To further increase robustness only the sign of $g_i - g_c$ is considered resulting in:

$$T = t(s(g_0 - g_c), \ldots, s(g_{P-1} - g_c)), \tag{2.40}$$

where

$$s(x) = \begin{cases} 1 & x \geq 0 \\ 0 & x < 0 \end{cases}. \tag{2.41}$$

To encode the extracted texture with a single descriptor a binary representation is used, where the value 2^i is assigned to each sign $s(g_i - g_c)$, i.e.:

$$LBP_{P,R} = \sum_{i=0}^{P-1} s(g_i - g_c)2^i. \tag{2.42}$$

This encoding results in 2^P possible values. Rotational invariance is achieved through a rotation scheme which aims to ensure that a maximal number of most significant bits, starting from g_{P-1}, are 0.

In order to reduce the number of possible patterns and exclude those patterns occurring infrequently the concept of uniform patterns is introduced. The uniformity of a pattern is based on the number of $0 \rightarrow 1$ transitions in the binary representation of a pattern. Only patterns with a uniformity score of 2 or less are used in the final representation, i.e.:

$$LBP^U_{R,P} = \begin{cases} \sum_{i=0}^{P-1} s(g_i - g_c) & \text{if } U(LBP_{P,R}) \leq 2 \\ P + 1 & \text{otherwise} \end{cases}, \tag{2.43}$$

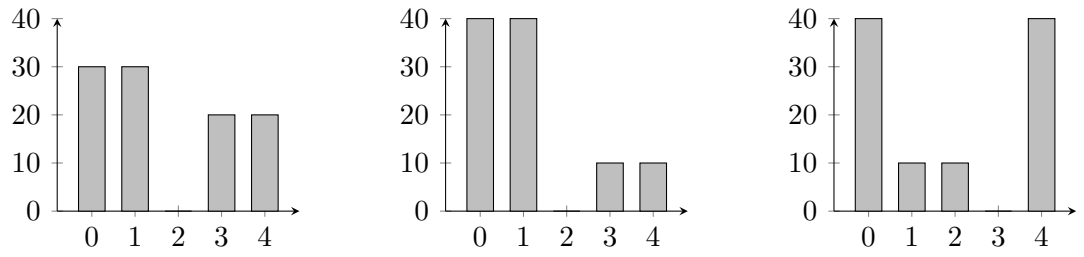
where the uniformity score is computed as

$$\begin{aligned} U(LBP_{P,R}) = &|s(g_{P-1} - g_c) - s(g_0 - g_c)| \\ &+ \sum_{i=1}^{P-1} |s(g_i - g_c) - s(g_{i-1} - g_c)| \end{aligned}. \tag{2.44}$$

The result of this is that the P uniform patterns are mapped to bins 0 through P while the remaining patterns are mapped to bin $P + 1$. The reason non-uniform patterns are discarded is that they make up only a small portion of the observed patterns and it is therefore hard to obtain accurate statistics for them. To obtain the actual histogram representation of an image's texture, the $LBP^U_{P,R}$ value is computed for every pixel in the image.

2.5.2 3D laser point clouds

3D point cloud data is becoming more common, mainly due to the Kinect, an affordable structured light sensor. However, the Kinect is restricted to short ranges and indoor environments. In outdoor environments 3D point cloud data is therefore gathered by rotating planar laser scanners or Velodyne laser scanners.



(a) Histogram 1 (b) Histogram 2 (c) Histogram 3

Figure 2.5: Examples of histograms. Histograms (a) and (b) have similar shape but differ in absolute value while histogram (c) has entirely different shape and values. Using Euclidean distance this notion of shape of the distribution is not captured, however, using Bhattacharyya distance this information is considered in the computation.

Common to the data provided by all of these sensors is that they provide 3D points with no direct information about the structure they represent, which can make their interpretation challenging.

Surface normal histograms

Data collected in urban environments is full of planar surfaces, which can be characterised by their normals. As such a histogram over the surface normals can be a good feature in such situations. Before we can compute a surface normal histogram we need to compute the normal of each point in the point cloud. As mentioned before, point cloud data contains no information about the structure these normals are estimating. Several methods exist which use different assumptions, an overview of common methods is given in Klasing et al. (2009).

Surface normals can be treated in a similar manner to the pixels in colour images, i.e. they are composed of three independent channels. This allows us to create one, two or three dimensional histograms. This representation discards a large amount of information such as distance and position. Furthermore, the histograms are extremely view point dependent. As such the histograms are not suitable for all applications, especially when classification is the final goal. With 3D point cloud data becoming more common, a large amount of descriptors suitable for recognition have been proposed, e.g.(Johnson and Hebert, 1999; Steder et al., 2011; Quadros et al., 2012).

2.5.3 Histogram distance

All the features discussed above create a histogram representation. In order to use these with the clustering methods described in the thesis we need to be able to compute the distance or similarity between pairs of histograms.

Algorithm 7: Earth Mover's distance

Input: Histograms a and b
Output: Distance between the histograms

$\mathsf{emd}_1 \leftarrow 0$
foreach $i \in \{1, \cdots, N\}$ **do**
  $\mathsf{emd}_{i+1} \leftarrow (a_i + \mathsf{emd}_i) - b_i)$
end
return $(\sum_i |\mathsf{emd}_i|)$

A simple way to obtain such a distance value is the Euclidean distance:

$$d_E(a, b) = \sqrt{\sum_i (a_i - b_i)^2}, \tag{2.45}$$

where a and b are histograms. This treats the histograms as simple high-dimensional vectors and ignores the fact that histograms are discrete probability distributions.

Compared to the Euclidean distance, the Bhattacharyya distance is designed to compare probability distributions and thus looks at their overlap. The distance is computed as follows:

$$d_B(a, b) = \sqrt{1 - \frac{1}{\sqrt{\sum_i a_i \sum_i b_i N^2}} \sum_i \sqrt{a_i b_i}}, \tag{2.46}$$

where N is the total number of bins. The result of this function is between 0 and 1 which makes it easy to work with.

Another popular distance metric is the so called earth mover's distance. The idea behind this metric is that it computes the amount of "dirt" that needs to be moved to transform one histogram into the other. As such similar histograms will have a lower cost then dissimilar ones. In the case of one dimensional histograms the earth mover's distance can be computed iteratively as shown in Algorithm 7.

The difference between these distances is best shown with an example. If we look at the histograms in Figure 2.5 it is clear that the histogram in (a) and (b) are more similar to each other then the other combinations. Looking at the distances computed by the three metrics in Table 2.1 the Bhattacharyya and earth mover's distance mirror this intuition clearly, while the Euclidean distance does less so. Additionally, for both the Bhattacharyya and earth mover's distance the distances $d(a, c)$ and $d(b, c)$ are almost identical which makes sense if we consider histograms (a) and (b) to be similar. This fact is not conveyed by the Euclidean distance. The main difference between the Bhattacharyya and earth mover's distance is that the former is directly normalised between 0 and 1 which can make its usage easier.

	Euclidean	Bhattacharyya	Earth Mover
$d(a, b)$	20.0	0.15	20
$d(a, c)$	37.4	0.44	220
$d(b, c)$	44.7	0.45	200

Table 2.1: Comparison of values obtained for different distance metrics on the example histograms.

2.6 Evaluation metrics

As the work in this thesis deals with clustering, commonly used measures such as accuracy are not ideal. For this reason metrics designed specifically to evaluate clustering results and outlier quality of points are needed. Some of these methods are introduced in the following.

2.6.1 V-Measure

Precision and recall based measures such as F-measure are not ideal for the evaluation of clustering results as we deal with multiple classes and pure classification accuracy is not the only important factor. For this reason V-Measure (Rosenberg and Hirschberg, 2007) is used which captures two properties of a solution, homogeneity and completeness, and combines them into a single value. Intuitively homogeneity encodes how uniform a cluster is while completeness captures how points belonging to one class are distributed over the clusters.

The metric is based on the concept of entropy which is used to compute the two measures. To compute the two values the method requires the true labels C of the N data points as well as the predictions in the form of the cluster assignments K. Based on this the contingency table $A = \{a_{ij}\} : i \in \{1, \ldots, |C|\}, j \in \{1, \ldots, |K|\}$ is built, which stores the number of points with class label i assigned to cluster j. The homogeneity value h is defined as follows:

$$h = \begin{cases} 1 & \text{if } H(C, K) = 0 \\ 1 - \frac{H(C|K)}{H(C)} & \text{otherwise} \end{cases} \tag{2.47}$$

where

$$H(C \mid K) = -\sum_{k=1}^{|K|} \sum_{c=1}^{|C|} \frac{a_{ck}}{N} \log \frac{a_{ck}}{\sum_{\tilde{c}}^{|C|} a_{\tilde{c}k}} \tag{2.48}$$

$$H(C) = -\sum_{c=1}^{|C|} \frac{\sum_{k=1}^{|K|} a_{ck}}{|C|} \log \frac{\sum_{k=1}^{|K|} a_{ck}}{|C|}, \tag{2.49}$$

while completeness c is computed as:

$$c = \begin{cases} 1 & \text{if } H(K,C) = 0 \\ 1 - \frac{H(K|C)}{H(K)} & \text{otherwise} \end{cases} \tag{2.50}$$

where

$$H(K \mid C) = -\sum_{c=1}^{|C|}\sum_{k=1}^{|K|}\frac{a_{ck}}{N}\log\frac{a_{ck}}{\sum_{\tilde{k}=1}^{|K|}a_{c\tilde{k}}} \tag{2.51}$$

$$H(K) = -\sum_{k=1}^{|K|}\frac{\sum_{c=1}^{|C|}a_{ck}}{|C|}\log\frac{\sum_{c=1}^{|C|}a_{ck}}{|C|}. \tag{2.52}$$

The final V-Measure score is then computed based on the homogeneity h and completeness c values as:

$$V_\beta = \frac{(1+\beta)hc}{(\beta h) + c}, \tag{2.53}$$

where the parameter β allows the measure to give more or less importance to the two parts. With $\beta < 1$ homogeneity is weighted more strongly, while for $\beta > 1$ completeness has more importance. A value of $\beta = 1$ gives equal importance to both parts and is therefore used throughout this thesis.

2.6.2 Local outlier factor

Local outlier factor (Breunig et al., 2000) (LOF) measures the outlier quality of a point p in a dataset D with respect to its surroundings. The metric is based on the notion of *k-distance*(p) which for any positive integer k is defined as the distance $d(p, o)$ where o is the k-th nearest neighbour of p. Based on this the *k-distance* neighbourhood of p, $N_k(p)$, is defined as the set of all points closer then *k-distance*(p), i.e.:

$$N_k(p) = \{q \in D \setminus \{p\} | d(p,q) < \textit{k-distance}(p)\}. \tag{2.54}$$

In order to reduce statistical fluctuations for points in close proximity to p a reachability distance is introduced as:

$$\textit{reach-dist}_k(p,o) = \max(\textit{k-distance}(o), d(p,o)), \tag{2.55}$$

which equals the *k-distance* for nearby points and the actual distance for distance points. As the goal is to detect density based outliers the local reachability density

is introduced as:

$$lrd_k(p) = 1/\left(\frac{\sum_{o \in N_k(p)} reach\text{-}dist_k(p,o)}{|N_k(p)|}\right), \tag{2.56}$$

which captures the local density of a given point p. Finally, to obtain the local outlier factor value for point p the local reachability density of p is compared with that of its *k-distance* neighbourhood, i.e.:

$$LOF_k(p) = \frac{\sum_{o \in N_k(p)} \frac{lrd_k(o)}{lrd_k(p)}}{|N_k(p)|}, \tag{2.57}$$

where k is the user defined number of other points to consider during the computations.

2.6.3 Jaccard index

The Jaccard index is a statistical measure that compares the similarity and diversity of sample sets and is defined as:

$$J(A,B) = \begin{cases} 1 & \text{if} \quad A, B = \emptyset \\ \frac{|A \cap B|}{|A \cup B|} & \text{otherwise} \end{cases}. \tag{2.58}$$

The value of Jaccard index is always $0 \leq J(A,B) \leq 1$.

Chapter 3

Self-supervised learning

3.1 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. 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 to learn 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 changes in the environment. Alternatively, methods that build a model of the environment in an unsupervised fashion have the advantage that they can adapt to changes more easily. Such methods also make robots more accessible to non-experts, as no special setup is required and allows operation in areas where no data is available for prior training.

In this chapter we present a method to learn the visual appearance of objects and whether or not they represent an obstacle to the robot. The entire learning process runs in real time and without the need for human supervision. We build the model using clustering, specifically, affinity propagation (Frey and Dueck, 2007), described in detail in Section 2.2. 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 (Zhang et al., 2008) allows us to cluster data streams in real time. To encode the visual appearance of the environment we use colour histograms and local binary patterns (Ojala et al., 2002). 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

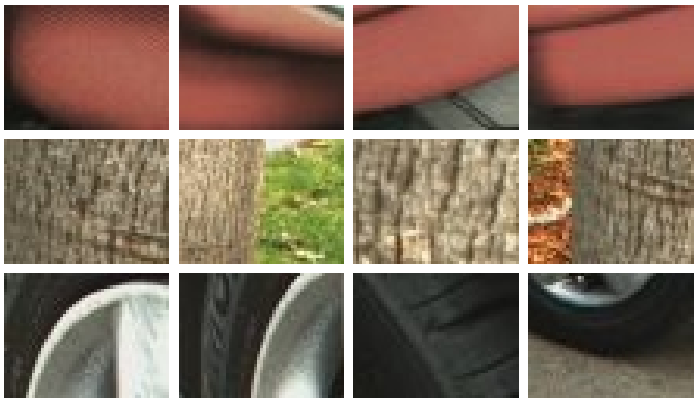


Figure 3.1: Examples of the visual appearance of object parts detected by our method. From top to bottom: seat of a chair, a tree trunk and a car tire.

on the labels obtained from the classifier a simple decision making procedure can determine safe motion commands for the robot. Building on this we propose an extension to affinity propagation, called meta-point affinity propagation, which aims to reduce the computational cost as well as offer robustness in the presence of noise.

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) (DARPA, 2004) 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 (Ng et al., 2001) and latent Dirichlet allocation (Blei et al., 2003). The work presented here follows similar ideas but it is the first to present a solution based on an online version of affinity propagation.

Using synthetic data we evaluate the properties of the different clustering methods. Experiments with a real robot show how the clusters capture the different objects in the environment accurately and how recognition of the different objects in the environment improves as time progresses. Additionally, we demonstrate that meta-point affinity propagation allows for long-term operation. Finally, we show that the proposed method can be easily adapted to other problems, such as learning to predict collisions based on laser scanner data.

The main contributions of this chapter are:

- a real-time system that learns a model of the appearance of objects in a totally unsupervised manner with streaming affinity propagation;
- a system that learns to recognise and predict collisions with the environment directly from laser scans without the need for human supervision;

- a clustering method based on affinity propagation which improves robustness and scalability in robotics applications;
- experimental evaluation in indoor and outdoor environments demonstrating that the robot learns to recognise obstacles through interaction.

The chapter is organised as follows. In Section 3.2 we present previous work related to our approach, before we present our method in Section 3.3. In Section 3.4 we present our extension to affinity propagation, meta-point affinity propagation, and evaluate the different methods in experiments in Section 3.5. Finally, we summarise the chapter in Section 3.6.

The work presented in this chapter was previously published in ICRA 2012 (Ott and Ramos, 2012a), ISER 2012 (Ott and Ramos, 2012b), and the IJRR special issue on long-term autonomy (Ott and Ramos, 2013b).

3.2 Related work

Traversability

In the context of the LAGR program several methods have been developed that learn to extract traversability information from images observed by a robot. Happold et al. (2006) predict the 3D terrain traversability from images based on data obtained from a stereo camera and a neural network classifier. They link colour features to geometry and use this to predict traversability with a histogram representation. The approach by Howard et al. (2006) 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. (2006) 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. (2006) require a supervised classifier to predict traversability, which requires data 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 beforehand 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 (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 laser point cloud, thus allowing the vehicle to predict terrain roughness before driving over it, such that the vehicle can slow down if needed. A similar approach was taken by Ulrich and Nourbakhsh (2000) 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 manner Maier et al. (2011) use the information of calibrated 3D laser scan and monocular vision to train an image based ground classifier which is then used to avoid obstacles in absence of continuous 3D data. The work by Modayil and Kuipers (2004) 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. Their approach mainly concentrates on the feature extraction while we focus on the model building in this work.

Vision

In the computer vision community, the topic of object detection has been extensively studied. 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.

Clustering

As indicated by the prior work in the LAGR project clustering is an important technique in unsupervised learning. The desired properties of a clustering method, however, are challenging. The properties include: speed, incremental updates, flexibility and, most importantly, automatic selection of the number of clusters. The work by Kim et al. (2006) for example employs a heuristic based clustering method to achieve this. Other principled methods include 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 the above methods make different assumptions when modelling the clusters and the way clustering is computed. DBSCAN for example assumes that clusters are based on density and closeness of points while spectral clustering assumes that a projection to a lower dimensional space affords better separability then the original one. Finally, Latent Dirichlet allocation assumes all observations can be modelled by a set of topics.

We are not the first attempting to use affinity propagation for real time applications. Streaming affinity propagation (Zhang et al., 2008) approaches this by representing clusters only through an exemplar and statistics about the cluster. Points which do not fit into the model are integrated once the entire model is recomputed. Other methods rely on heuristics to reduce the number of points involved in the computations. One such approach is presented by Katz et al. (2010). They use a subset of all the available points, based on the current clustering solution, when recomputing the clusters with new data. Xia et al. (2008) propose to run AP on a subsample of the data and assign the remaining points in a k-means fashion to the result. Finally, Furtlehner et al. (2010) introduce a hierarchical approach which shows reduction in complexity with little to no loss in accuracy. The original data is divided in multiple subsets which are clustered individually and then merged hierarchically using a modified version of affinity propagation called weighted affinity propagation. They shown that this reduces the computational complexity to $O(N^{(h+2)/(h+1)})$, where h is the depth of the hierarchical strategy. In comparison to these methods we approach the problem from a different angle. We directly reduce the number of points by merging them into meta-points. As we will see in the experiments this has a significant impact on the runtime with negligible clustering quality degradation.

3.3 A model to avoid obstacles

In this section we describe a system to enable 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 whether they represent an obstacle to the robot. Features are clustered with a combination of affinity propagation and streaming affinity propagation. Streaming affinity propagation 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

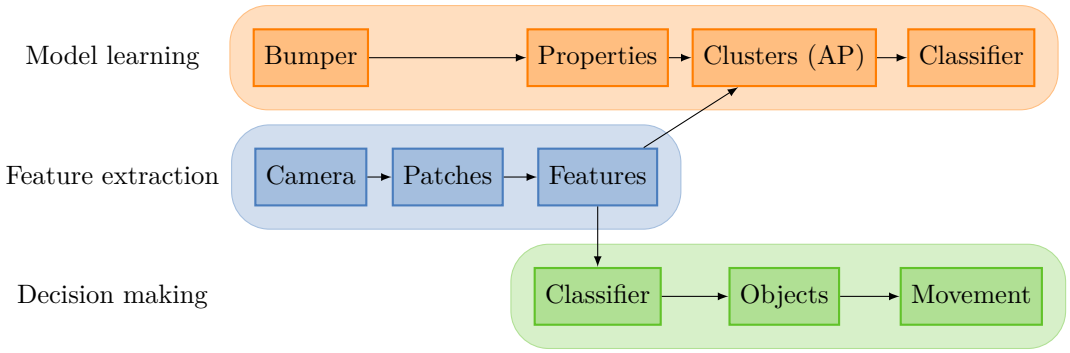


Figure 3.2: Overview of the processes of our system. Common to all operations is the division of the image into patches and the extraction of features from these patches. The learning of a model then proceeds by associating the obstacle property captured by the bumper to these features. Next the features are clustered using affinity propagation and the result is used to build a classifier. When using an existing model for motion decisions each patch is classified according to its features and the assigned cluster and traversability properties are obtained. This information is then used to determine the best motion command for the robot.

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 interaction with the environment, i.e. collisions or free motion. A short overview of the system is provided next.

3.3.1 Overview

A schematic overview of the processing pipeline is shown in Figure 3.2. In the first step our method extracts features from raw images (centre row in Figure 3.2) in the following manner:

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 learn a model of the environment as follows (top row in Figure 3.2):

1. assign to each patch a traversability property obtained from the bumper for object classification;

2. add 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 neighbours classifier to classify new observations as either traversable or non-traversable.

With a model of the environment we can make decisions about the motion commands the robot should execute using the following approach (bottom row in Figure 3.2):

1. obtain the object class for the features extracted from the image patches by classifying them using the k-nearest neighbours 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.

3.3.2 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 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. This approach is simple to implement and use while also being computationally efficient, important properties for a real-time system such as the one we describe. In 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 computation while providing a reasonable approximation when the individual patches are small enough. Different number of patches have been tried and values from 9 to 144 worked equally well. However, if the patches become too small the performance will degrade. More elaborate approaches, such as watershed based methods (Meyer and Beucher, 1990) or super pixel methods (Felzenszwalb and Huttenlocher, 2004; Achanta et al., 2012) might provide better approximations but are also computationally more expensive. For each of the patches we compute two different histogram features: the colour distribution in the HSV colour space, and the distribution of local binary patterns Ojala et al. (2002). These features allow us to capture both colour 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 (see Section 2.5.3). This allows us to treat the features as distributions rather then simple vectors.

3.3.3 Obstacle label extraction

In order to associate obstacle information with the learned objects we need to know how the robot interacts with the environment. Therefore, whenever the robot collides with obstacles in the environment we label the currently observed image patches as obstacles. 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.

3.3.4 Building the model

Features extracted from image patches are added sequentially into the clustering system. The clustering is performed by a combination of affinity propagation (AP) and streaming affinity propagation (STRAP) which are explained in detail in Section 2.2.

The pseudo code in Algorithm 8 shows the steps performed for each observation we add. Each observation is added to the long-term clustering instance $\mathcal{M}_{\mathsf{long}}$ (STRAP), where it is either used to update an existing cluster or added to the outlier reservoir, $\mathcal{M}_{\mathsf{long}}.\mathsf{reservoir}$. In the latter case, the data point is additionally added to the short-term clustering instance $\mathcal{M}_{\mathsf{short}}$ (AP), which is rebuilt thereafter. Once the outlier reservoir is full $\mathcal{M}_{\mathsf{long}}$ is rebuilt and information from $\mathcal{M}_{\mathsf{short}}$ is merged into $\mathcal{M}_{\mathsf{long}}$.

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:

$$p(\text{obstacle}_i) = \frac{|\text{obstacles}_i|}{|\text{obstacles}_i| + |\text{non-obstacles}_i|}\,, \tag{3.1}$$

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 obstacle and non-obstacle respectively. Note that the clustering is based purely on the visual features and the obstacle properties are never used for this.

Algorithm 8: AP & STRAP integrating a novel observation

Input: Observation obs, STRAP instance $\mathcal{M}_{\text{long}}$, AP instance $\mathcal{M}_{\text{short}}$
Output: Updated $\mathcal{M}_{\text{long}}$ and $\mathcal{M}_{\text{short}}$ instances

```
1  Insert obs into M_long
2  if obs ∈ M_long.reservoir then
3  |  Insert obs into M_short
4  |  Recompute M_short
5  end
6  if Count(M_long.reservoir) > max_count then
7  |  Recompute M_long
8  |  Merge M_short into M_long
9  |  Reset M_short
10 |  M_long.reservoir←∅
11 end
12 return M_long, M_short
```

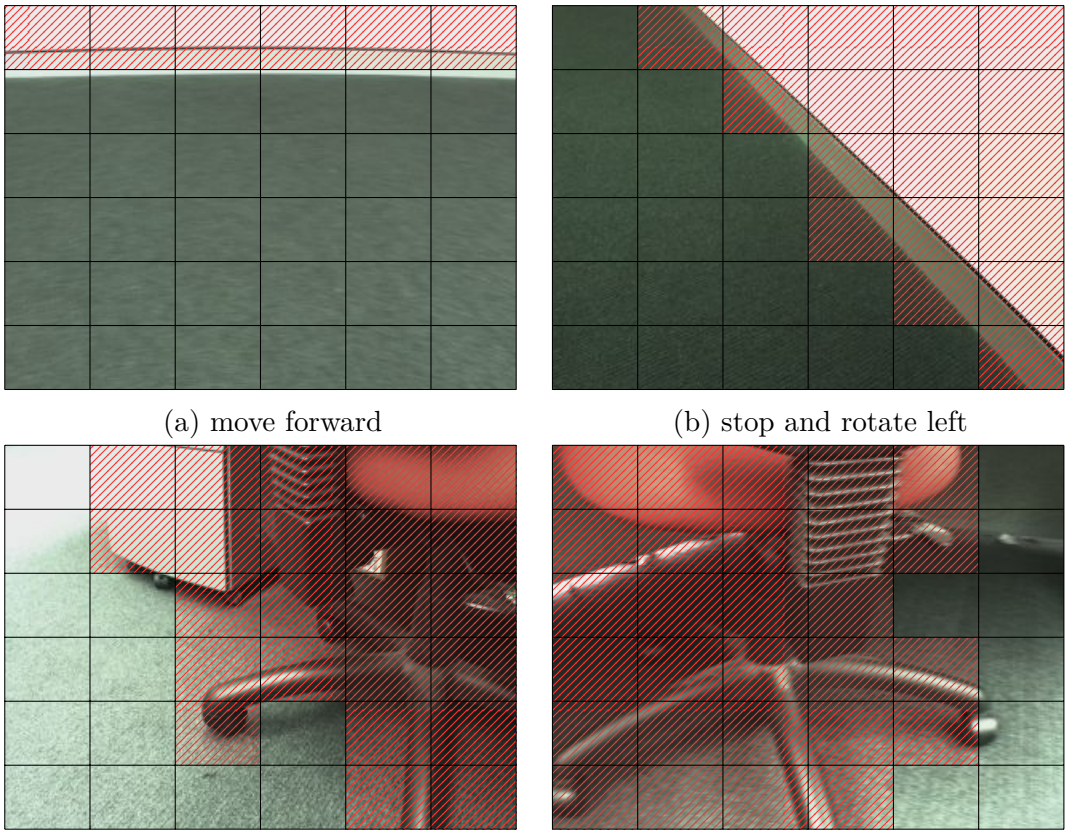
There is always the possibility that one visual appearance represents an obstacle in one environment but not in another. In such a case the model will adapt the obstacle probability of a cluster over time by making new observations. This is not ideal as we try to model two different objects as the same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

3.3.5 Building the classifier

In order to predict where obstacles are located in new images a classifier is trained based on the exemplars of the current model. As the system has to run in real-time and the model can change frequently, classifiers that are computationally expensive to train can not be used. For this reason we use a k-nearest neighbours classifier which can be efficiently trained directly 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.

3.3.6 Decision making

The first step of the decision making process is to label the current image with obstacle information. This is performed by classifying all patches using the previously trained classifier. This yields an object class and the associated obstacle property for each patch in the image. Based on the arrangement of the obstacle patches in the image a simple set of heuristics decides how the robot should move. The heuristics provide three types of instructions to the robot:



(a) move forward (b) stop and rotate left

(c) stop and rotate left (d) rotate until no longer stuck

Figure 3.3: Exemplary classification results and movement decisions. Obstacles are denoted by the shaded areas while the command decision is listed below each image.

- continue on the current path;
- stop and rotate left or right;
- stop, select new destination, and turn on the spot.

These actions are selected based on the classification result of the current image. The first action type is used when there is enough free space for the robot to continue moving on its current heading, as shown in Figure 3.3a. In case that there are obstacles to either the left or right of the robot the second action is used to prevent collision with the environment, as can be seen in Figures 3.3b and 3.3c. Finally, if the robot is surrounded by obstacles it discards its current goal and selects a new one. Then it rotates on the spot until a heading is found that allows the robot to move freely again, an instance of this can be seen in Figure 3.3d.

3.4 Meta-point affinity propagation

Affinity propagation requires a similarity matrix S which contains the pairwise distances between all N points in the dataset. Based on this similarity matrix the solution is computed by iteratively updating the availability α_{ij} and responsability ρ_{ij} messages, defined as:

$$\alpha_{ij} = \begin{cases} \sum_{k \neq j} \max(0, \rho_{kj}) & i = j \\ \min\left[0, \rho_{jj} + \sum_{k \notin \{i,j\}} \max(0, \rho_{kj})\right] & i \neq j \end{cases}, \tag{3.2}$$

$$\rho_{ij} = s_{ij} - \max_{k \neq j}(s_{ik} + \alpha_{ik}). \tag{3.3}$$

Computing these messages requires quadratic runtime in the number of data points which is prohibitively slow for large data sets. To reduce the computational burden of AP the number of data points has to be reduced.

The system proposed in the previous section used streaming affinity propagation (Zhang et al., 2008) to handle the continuous image stream. STRAP is capable of handling data streams in real-time by not storing data points but only statistics about clusters. Each cluster is represented by a 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. In addition to these cluster statistics there is a fixed size storage for points that do not fit into the current clusters. New points are used to either updated the cluster statistics or placed in the storage if they are too far away from the centre of the clusters. Once the storage is full the entire clustering solution is updated. During the update the statistics about the clusters are used generate similarity matrix entries. The result is that AP is run on a small similarity matrix and with reduced frequency. However, this design has two drawbacks: (i) there is some delay between novel observations being made and them being represented by the clusters and (ii) clusters that have not been observed for some time can be "lost", by merging them with other clusters.

These two drawbacks make it difficult to use in long-term autonomy scenarios and we propose a different method to increase the speed of affinity propagation which guarantees that we will not forget information. Our method, called meta-point affinity propagation, 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

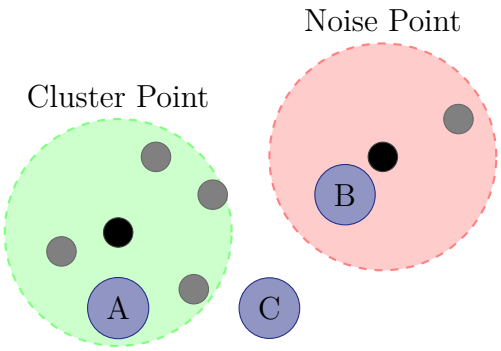


Figure 3.4: Visualisation of meta-points and the different cases that can occur when adding a new data point. A is merged into the cluster-point while B is merged into the noise-point. Finally, C is used to create an entirely new meta-point.

reduce the number of points involved when performing affinity propagation without running the risk of erasing knowledge.

Formally a meta-point is a 4-tuple $P_i = (c_i, \mu_i, e_i, t_i)$ and stores the number of points c_i represented by a meta point, the mean μ_i of all represented points, the exemplar e_i of the meta-point, and the last time t_i a point was added to the meta-point. Besides the immediate effect of reducing the computational burden, the concept of meta-points has two additional benefits:

1. the number of points to be clustered depends on the size of the feature space and not the number of observations;

2. random observations can be dealt with in a straight forward manner, by ignoring meta-points representing few actual points.

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, after a while the meta-points will cover the entire feature space and all new observations will be mapped to an existing meta-point. The second point requires us to distinguish between two types of meta-points: 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 pseudo code in Algorithm 9 shows the steps performed in order to add a point p into either the set of cluster-points $\mathcal{P}$ or the set of noise-points $\mathcal{N}$. 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 visualised in Figure 3.4. This method requires two parameters: $\theta_{\text{min-points}}$ a threshold which decides if a meta-point i is a cluster-point ($\theta_{\text{min-points}} \geq c_i$) or a noise-point ($\theta_{\text{min-points}} < c_i$). The second parameter $\theta_{\text{similarity}}$ 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. $\theta_{\text{similarity}}$ is more complicated as it is dependant on the magnitude of the feature space. We typically use a value between 5 to 15% of the magnitude of the feature space.

We use two distinct sets for noise and cluster meta-points to both make their difference more prominent and 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 sensor or systematic noise, as these produce consistent and continuous observations.

3.5 Experiments

In the following several experiments on synthetic as well as real data are presented. Synthetic data is used to compare the properties of AP, MPAP and STRAP on clustering problems. Next, we show how AP and STRAP can be used to learn an environment model using the method outlined in Section 3.3. The same data is also processed using MPAP, building a model capable of explaining data collected months later. Finally, we present another application which allows a robot to predict collisions based purely on laser range data without any knowledge about the robot's shape using the pipeline shown in Section 3.3.

Algorithm 9: MPAP add new observation

Input: New data point p, Data point set $\mathcal{P}$, Noise point set $\mathcal{N}$
Output: Updated sets $\mathcal{P}$ and $\mathcal{N}$

$\mathsf{nn} \leftarrow \min_{q \in \mathcal{P}} \mathtt{Dist}(q, \mathsf{p})$
if $\mathtt{Dist}(\mathsf{nn}, \mathsf{p}) < \theta_{\mathsf{similarity}}$ **then**
  $\mathsf{nn} \leftarrow \mathsf{nn} \cup \mathsf{p}$
else
  $\mathsf{nn} \leftarrow \min_{q \in \mathcal{N}} \mathtt{Dist}(q, \mathsf{p})$
  if $\mathtt{Dist}(\mathsf{nn}, \mathsf{p}) < \theta_{\mathsf{similarity}}$ **then**
    $\mathsf{nn} \leftarrow \mathsf{nn} \cup \mathsf{p}$
    if $|\mathsf{nn}| \geq \theta_{\mathsf{min\text{-}points}}$ **then**
      $\mathcal{P} \leftarrow \mathcal{P} \cup \mathsf{nn}$
      $\mathcal{N} \leftarrow \mathcal{N} \setminus \mathsf{nn}$
    end
  else
    $r \leftarrow$ Create new meta-point from p
    $\mathcal{N} \leftarrow \mathcal{N} \cup r$
  end
end
return $\mathcal{P}$, $\mathcal{N}$

3.5.1 Speed and quality evaluation on synthetic data

In these synthetic experiments we investigate both quality and speed of AP, STRAP and MPAP on data sampled from random 2D Gaussian distributions with randomly selected mean μ and covariance Σ. The noise is sampled uniformly over the area of the data. The similarity between points is computed using the euclidean distance. We perform comparisons on data with and without noise. Exemplary results are shown in Figure 3.5. The top row shows the results on noise free data while the bottom row has added noise. In the noise free case all methods produce similar results, however, when noise is added we see that AP splits some of the clusters while STRAP generates some extremely large clusters. MPAP on the other hand performs just as it did before as it only deals with the meta-points representing enough raw points, discarding the noise points.

In a quantitative analysis we performed 1000 trials with both noisy and noise free data and compared the runtime, V-Measure, and number of points clustered by each of the methods. The results in Table 3.1 show that MPAP and STRAP are significantly faster then AP. In the noise free case MPAP and STRAP have a better V-Measure score then AP, however, this can be largely ignored as MPAP and STRAP label only a fraction of the points while AP labels all points. Looking at the results for data containing noise we can see that the speed differences stay similar. The big change is in the V-Measure score where both AP and STRAP

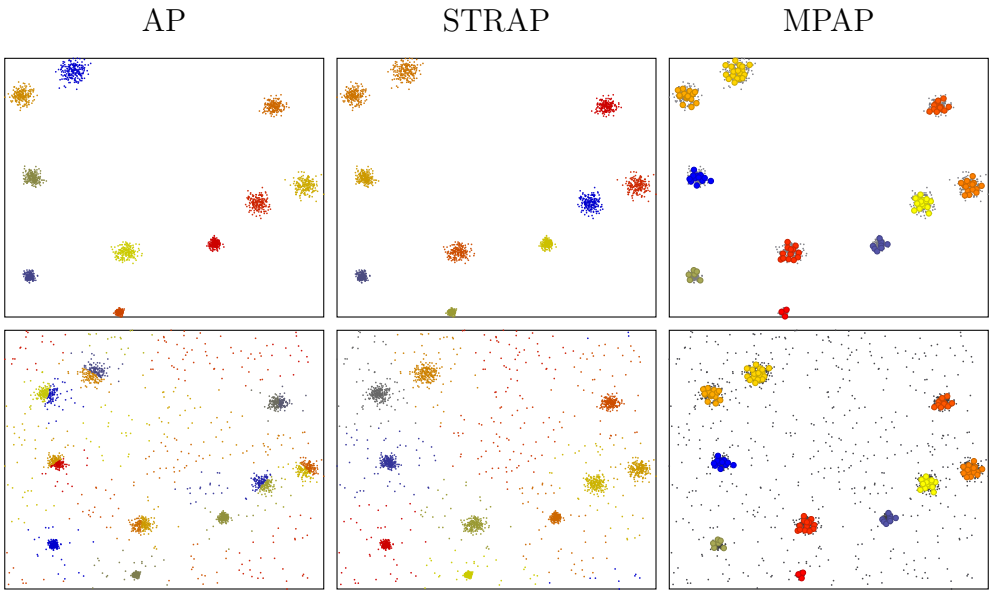


Figure 3.5: Results obtained with AP, STRAP and MPAP on 2D data. The first row shows results obtained on noise free data, while the second row shows the results on the same data with added noise. The colouring of points indicates the cluster assignments. In case of MPAP the larger circles indicate the meta points.

drop as they try to explain every point, including the noise. MPAP ignores the noise as the meta-points representing noise are ignored and therefore there is no loss in clustering quality.

To further demonstrate the need for MPAP we show the effect data ordering has on STRAP. We run STRAP and MPAP with the same data once randomly shuffled and once ordered by clusters. As Table 3.2 shows, if the data is ordered STRAP performs poorly. This is the effect of forgetting we mentioned earlier and is caused by the way the clustering update uses the data. MPAP does not suffer from this, as similar points are simply merged into a single meta-point which will never be removed.

From these experiments on synthetic data we can see it is critical to improve the performance of AP, as its runtime is $O(N^2)$ where N is the number of points clustered. Both STRAP and MPAP significantly improve on AP's runtime, however, STRAP can perform badly when the data stream is not diverse enough, something that can easily occur in robotics.

3.5.2 Visual appearance learning and obstacle avoidance

In this section we present evaluation results of our system, presented in Section 3.3, 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 per-

	Time (s)	V-Measure	Points
	No Noise		
AP	14.61 ± 2.92	0.87 ± 0.18	2000
STRAP	0.02 ± 0.01	0.94 ± 0.08	2000
MPAP	0.08 ± 0.03	0.94 ± 0.13	195 ± 62
	With Noise		
AP	22.26 ± 3.48	0.73 ± 0.12	2200
STRAP	0.02 ± 0.01	0.86 ± 0.04	2200
MPAP	0.07 ± 0.02	0.92 ± 0.06	161 ± 41

Table 3.1: Results for 1000 runs of AP and MPAP on synthetic 2D data with varying numbers of clusters. V-Measure was computed based on the points used in the clustering. MPAP achieves better results then AP and is significantly faster since it processes fewer points.

	Data Ordering	
	Random	By Clusters
STRAP	0.95 ± 0.08	0.65 ± 0.11
MPAP	0.96 ± 0.06	0.96 ± 0.06

Table 3.2: Impact of data ordering on STRAP and MPAP. Unless the data stream is diverse enough the quality of the clustering suffers with STRAP while MPAP has no such issues.

formed 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 trigger 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 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 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.

Learning visual appearance from images 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 no information about the environment available. As time progresses previously unobserved parts of the environment are encountered and their appearance and

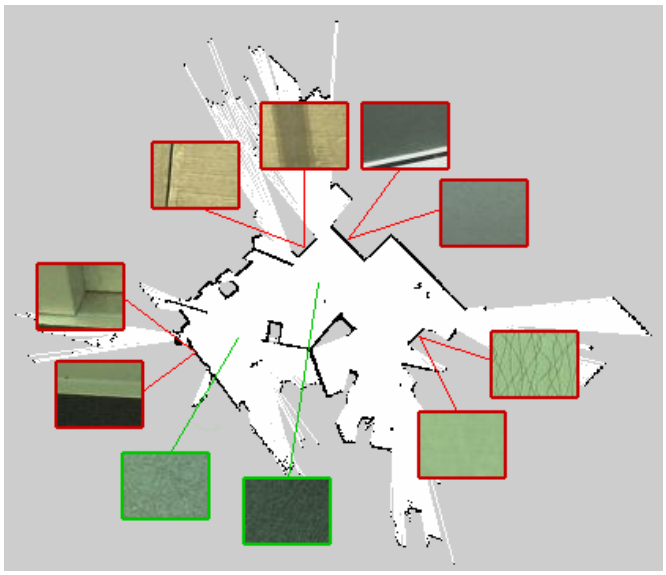


Figure 3.6: Map of one of the indoor environments with exemplary image patches referenced to the location they have been perceived at. The coloured box around the patches also indicates if they represent an obstacle (red) or free space (green).

obstacle property is added to the model.

In an initial experiment the robot moved for 15 minutes in different environments, both indoor and outdoor, colliding while gathering images. The indoor environments are typical office areas with desks, cubicle walls, chairs, etc. The outdoor environment covers a grass area and a parking lot with variability in the illumination due to shadows from nearby trees. While the robot travels through these environments the model was continuously updated. One would expect that at the beginning all observations are new to the system and thus require incorporating into the model. Over time, as more of the environment is covered, fewer observations contain new information. This expected behaviour is verified by plotting the percentage of observations that were novel to the system over time in Figure 3.7. We can see that for all three environments the first few minutes, when the robot is still mostly traversing previously unvisited 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. Sometimes observations contain information explained by the robot looking at areas that have not been observed or changes in the lighting conditions producing observations that are significantly different from what the model predicts.

A visual representation of the model learnt in an indoor environment is shown in Figure 3.6. The grid map is not part of the model but is shown for visualisation purposes. The patches show the appearance the robot perceives at those

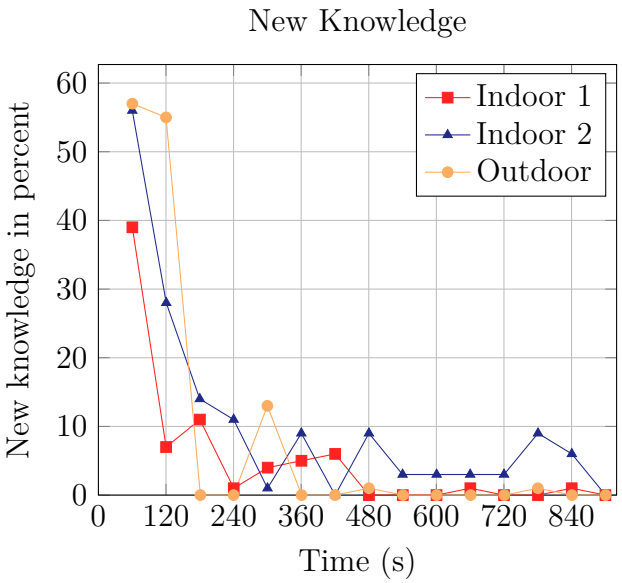


Figure 3.7: The plot shows the percentage of all observations made during a 60 s window that add new information to the environment model. It is clearly visible that the majority of observations which lead to new knowledge are observed in the beginning.

locations. A green border indicates a non-obstacle while a red border indicates an obstacle. We can easily see how the carpet floor is recognised as not being an obstacle while other things such as walls, doors and boxes are selected and correctly labelled as obstacles.

Exemplars found by the clustering algorithm in both indoor and outdoor scenarios are shown in Figure 3.8. As can be seen the clusters found can be easily distinguished from each other and represent the different types of objects present in the environment, such as floors, pavement, walls, and predominant obstacles. Ideally, clusters should be distinct from each other, i.e. separated in the feature space. However, clusters should also contain a reasonable amount of data points, i.e. allow for a certain amount of variability within a cluster. The examples of cluster members shown in Figure 3.9 demonstrate that the clusters obtained exhibit this property. Each row in Figure 3.9 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.

Figure 3.8: Examples of the exemplars as determined by streaming affinity propagation. (Left) Results from indoor experiments. (Right) Exemplars obtained in outdoor experiments.

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 recognised and stopped at a distance of 0.94 m $\pm$ 0.23 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.

Model consistency for long-term autonomy

The previous experiments have shown how STRAP and AP can build a suitable model. Here, we demonstrate that using MPAP we can build a model in the same way without the drawbacks of STRAP. As we are interested in long-term autonomy we also investigate how the model handles data that has been collected some time apart. To demonstrate this the robot moved between two visually distinct outdoor environments. Figure 3.10 shows exemplary image patches from the areas used in the experiment. The areas cover a parking lot, grass area, and a courtyard with the illumination varying from sun lit to overcast between the various observations. The robot starts out in environment A and stays there for 10 min before moving to environment B from where it returns to area A after 10 min. The plot in Figure 3.11 shows the percentage of observations made during a 60 s window which are considered novel. The two vertical lines in the plot indicate the transition between the two environments. At the first transition there

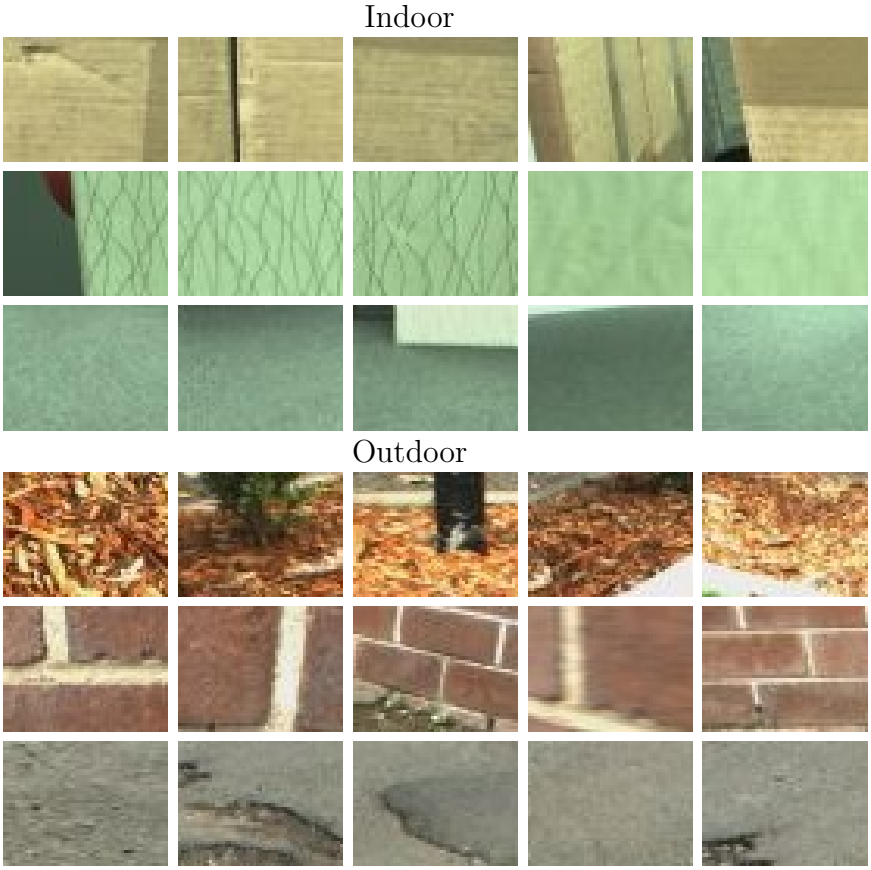


Figure 3.9: Examples of cluster members from both indoor and outdoor experiments. Each row contains image patches that are assigned to the same cluster. The examples show that even if the appearance changes significantly between images, the clustering procedure is still able to assign them to the appropriate cluster. The rows from top to bottom represent a cardboard box, a piece of structured room divider, carpet, wood chips, brick wall and asphalt.

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 extended period of time the above experiment is repeated with data gathered several months after the initial visit of area A, referred to as A'. We perform the same task as in the previous experiment, moving from A to B and then to A'. Figure 3.12 shows the plot of observation novelty over time. We can see a behaviour very similar to that in Figure 3.11. We have spikes in the novelty when first visiting area 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 the majority of the existing model is reused, even though the model is based on data gathered months before. While this experiment is not performed with a robot operating for an extended period of time it shows that the method can deal with changes that occur over time by reusing an existing model

Figure 3.10: Exemplary images from the different areas, from top to bottom A, A' and B. Area A and A' differ mainly in illumination and camera setting changes which affects the colour appearance.

	Preprocessing	Feature extraction	Insert data	Cluster data
Overall	3.7 s	5.3 s	47.9 s	1.0 s
Per Image	2.5 ms	3.6 ms	32.5 ms	0.7 ms

Table 3.3: Breakdown of the runtime of meta-point affinity propagation when processing the long-term autonomy dataset. Time for the entire process, as well as per image is shown.

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.

This can be executed in real time on a robot, as the number of points to be clustered is typically less then 1000 which can be clustered in a second. The most expensive operation over time is the addition of points into the model. However, adding a single point is very cheap. The initial 1472 images are split into 52 992 patches from which 873 meta-points are formed. A breakdown of the runtime of the entire system is given in Table 3.3. The processing time for a single image shows that images can be processed a 30 Hz using a current Intel Core-i5 processor. Furthermore, since the original images are not stored the entire method is CPU bound.

3.5.3 Learning to predict collisions from laser data

In this section we 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 any knowledge of the robot's physical dimensions or location of the sensor. It is purely based on the laser scanner returns and the information of the bump sensor on the robot. The

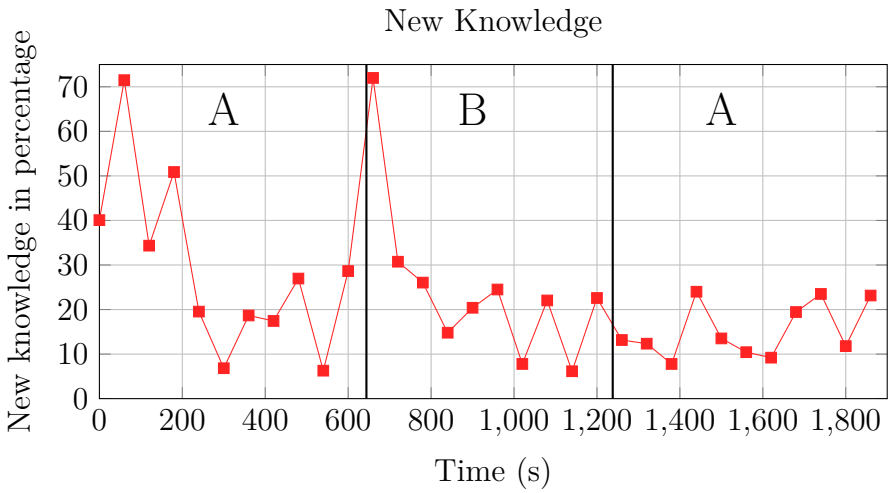


Figure 3.11: This plot shows the percentage of novel observations when the method is presented with two types of environments starting in environment A then moving to environment B and then back to environment A. The black vertical lines indicate the transition from one environment into the other. This clearly shows two desirable properties: upon entering a new environment most of the observations are novel and need to be represented, when switching back into the already known environment A no such spike in novelty occurs.

processing pipeline of this method, shown in Figure 3.13, is very similar to the one used in Section 3.3.1 and 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.

Model overview

The first step of the system is to extract features from the raw laser scanner data. To this end each laser scan is divided into bins $\mathcal{B}^k$ containing a fixed number of sequential readings, i.e.:

$$\mathcal{B}^k = \left\{ S_i : i \in \{ck, \ldots, c(k+1) - 1\} \right\} \quad \text{with} \quad c = \frac{|\mathcal{S}|}{|\mathcal{B}|}, \tag{3.4}$$

where $\mathcal{S}_i$ is the i-th laser range value, $\mathcal{B}^k$ the k-th bin, $|\mathcal{S}|$ the number of range readings, $|\mathcal{B}|$ the total number of bins, and c the number of readings per bin. From each bin a single feature value is computed which results in a feature vector or histogram representing the entire scan which are then clustered using meta-point affinity propagation. The information of the bumper, i.e. collision or no collision, is associated with each scan and accumulated in the clusters, which

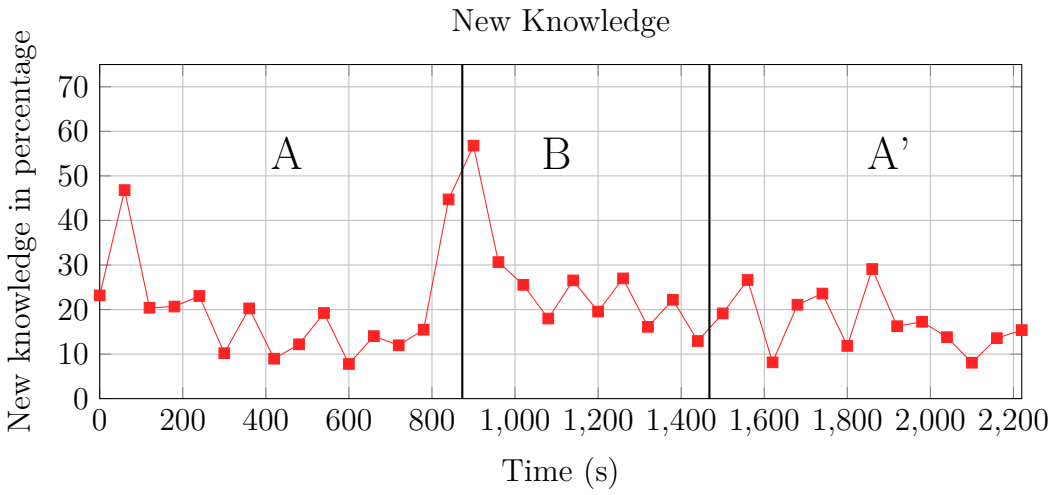


Figure 3.12: This plot shows the percentage of novel observations when the data from environment A and A' is recorded several months apart, emulating long-term operation. From the graph we can see that although the data was recorded several months apart the method still is able to reuse most of the existing model when visiting A'.

allows the computation of a collision probability for each cluster. Based on the cluster exemplars we build a simple k-nearest neighbours classifier. In order to decide if a robot is about to collide with the environment we compute the feature vector of the current scan and classify it using the classifier. The classifier output tells us whether the observation indicates a collision or not, which allows us to stop the robot if needed.

Feature extraction

The feature design is important as it will allow us to obtain a succinct representation of the entire laser scan. The goal in our case is a feature that is capable of capturing the broad shape of the laser scan which should be enough to distinguish different types of observations. In the following we show the different functions used to extract a single value from each bin $\mathcal{B}^k$ to produce the histogram representation of a scan:

- Min-value: $\min(\mathcal{B}^k)$
- Max-value: $\max(\mathcal{B}^k)$
- Mean-value: $\frac{1}{|\mathcal{B}^k|}\sum_i(\mathcal{B}_i^k)$
- Global difference: $\max(\mathcal{B}^k) - \min(\mathcal{B}^k)$
- Sequential difference: $\frac{1}{|\mathcal{B}^k|-1}\sum_1^{|\mathcal{B}^k|-1}(|B_i^k - B_{i+1}^k|)$

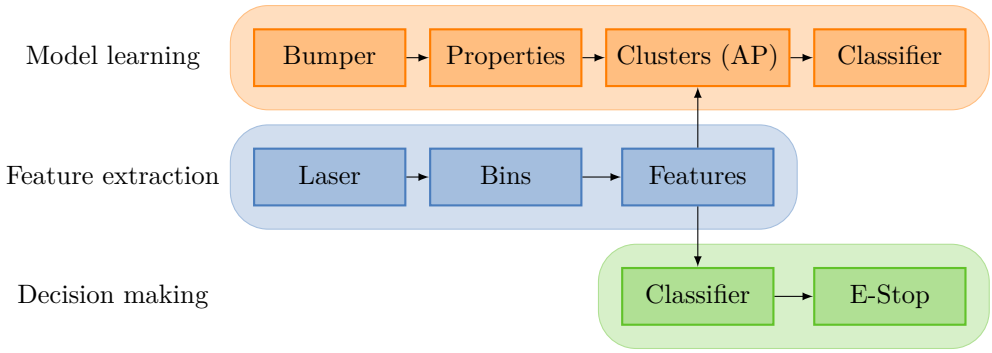


Figure 3.13: Overview of processing pipeline used to predict collisions based on raw laser data. We extract features from the binned laser scans which are then either used to build the model or make decisions. Building the model involves clustering the features and attaching the obstacle or non-obstacle label obtained from the robot's bumper. Based on this clustering result the classifier is built. The decision making uses the classifier output for new features to decide if the robot needs to stop.

where $\mathcal{B}_i^k$ is the range value of the i-th entry in the bin. The first three features capture straight forward statistical information. The last two are more complex. The global difference feature tells us how much the distance changes over the entire bin, independent of location. The sequential difference features gives an intuition of 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, see Section 2.5.3.

Classification performance

We evaluate how well the different features perform at grouping laser scans and predicting if they correspond to the robot colliding with the environment. For this we collected datasets in which the robot collides with the environment in both indoor and outdoor environments. The main difference between those is that the indoor environment has lower 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 3.14 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

	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

Table 3.4: Table showing the area under (ROC) 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 fails in the outdoor environment.

are distinct differences between collision and collision-free scans. In particular the amount of max-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.

We evaluate the different features with two different bin sizes, 6 bins with 30 values each and 12 bins with 15 values each. Figure 3.16 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 6 bins. As for the features themselves: the "min-value" feature performs best, however, only if we accept a large percentage of false positives. Both the "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 the "max-feature" performs so well can be explained by the large number of max-range readings produced by the laser scanner as shown in Figure 3.14. 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 similar 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 3.4 and paint a similar picture. It is easy to see how similar 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 for the task of collision prediction.

Collision prediction

Another way to look at how well the learned model performs is by comparing the timeline of collision and non-collision scans. Figure 3.15 shows the plot obtained for the indoor dataset with the "global-distance" feature using a threshold of 0.2. The top plot in red shows the hand labelled 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.

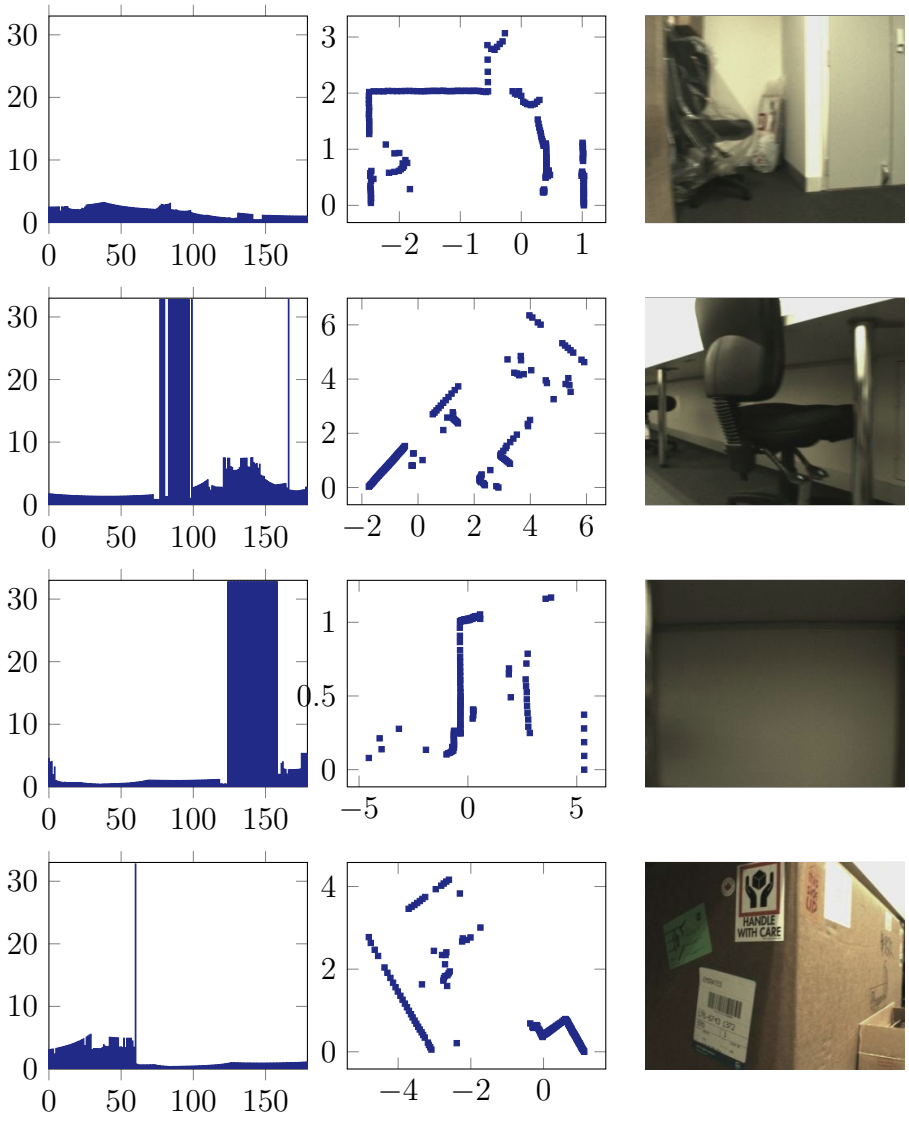


Figure 3.14: Some exemplary scans shown in both polar and Cartesian coordinates as well as an image taken by the robot. The first two rows represent cases where no collision with the environment is present. The second two rows both contain collisions. The third row is in collision with a metallic table leg to the left while the fourth row is in collision with the big cardboard box in front of the robot.

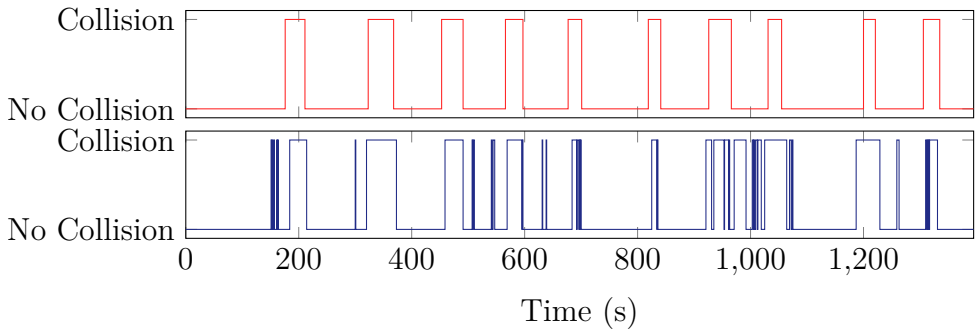


Figure 3.15: This figure shows the classification of each single scan as it is received by the robot as it travels through an environment. The top plot shows the ground truth labels and the lower plot shows the prediction made by our method.

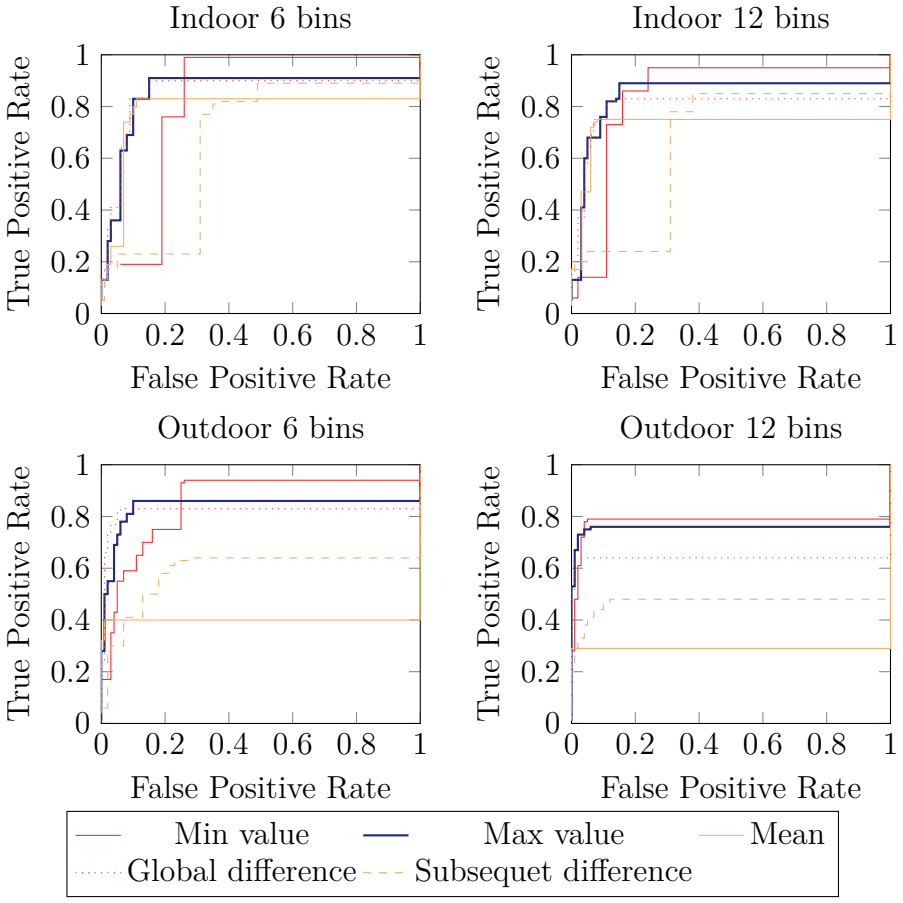


Figure 3.16: ROC plot for different features evaluated on different binning sizes of laser scans in indoor and outdoor environments. We can see similar trends independent of the number of bins we use. The "min value" feature obtains the best results, however, at a larger false positive rate then other features. Both the "max-value" and "overall difference" features obtain similar results.

3.6 Summary

We developed an online method to build a model of the environment without the need for human supervision. This is achieved by clustering the visual appearance of image patches observed by a robot using a combination of AP and STRAP. Additionally the information obtained by the interaction of the robot with the environment is used to assign the discovered objects with an obstacle property. To increase robustness for long-term operation we proposed meta-point affinity propagation which overcomes drawbacks of streaming affinity propagation, namely the loss of information over time

Using AP and STRAP we show how the proposed system is capable of learning an accurate model which allows the robot to recognise objects and avoid obstacles. We then demonstrate how MPAP can build consistent models capable of recognising already visited areas even if the observations are several months apart. Finally, we show how the proposed pipeline can be used to learn to predict collisions based on raw laser scanner data without knowledge about a robot's configuration.

The presented method is one of the first attempts to address the problems of long-term autonomy and model building with incremental unsupervised and self-supervised learning techniques. We believe this topic has a lot of potential for future work both algorithmically and experimentally.

Chapter 4

Layered clustering

4.1 Introduction

In the previous chapter we have shown how to learn representations of the environment using only the robot's sensory information. However, only a single sensor was used, whereas robots normally have multiple different sensors. Therefore, it is desirable to cluster the information of multiple sensors together. Typically, this involves the construction of a function that weights features extracted from the different sensors against each other. Constructing and tuning such a function is non trivial and typically does not generalise well. For this reason we present a novel algorithm that automatically clusters data from multiple sensors. The algorithm only requires the features to be extracted on a per sensor basis, combining the features is done automatically by the proposed method.

The presented method is called layered affinity propagation (LAP) and, as the name suggests, is an extension to affinity propagation (Section 2.2). The term layered refers to the fact that we represent the similarity values derived from each data source and the associated features by a separate affinity propagation instance or layer. Broadly speaking our method consists of the following two types of layers:

1. data layers, each representing a single data source;

2. a merging layer, ensuring an overall consistent clustering solution.

During clustering the data layers are all updated independently followed by an update of the merging layer. This cycle is repeated until convergence is achieved. Because we do not explicitly define how the different features are merged, we can use simple, well known methods to obtain similarities between data points for a single data source. The difficult part, which is to decide how to best combine the different data sources, is left to the algorithm.

The main contribution of this chapter is a novel, principled way to perform clustering of multiple data sources by message passing. In experiments we show that the method is capable of handling data from different sensors with significantly better clustering quality than alternative methods. A first set of exper-

iments demonstrates the capability to perform scene segmentation on RGB-D data collected indoors with a Kinect sensor. In a second experiment we cluster segments extracted from a Velodyne and camera combination in an urban setting into groups of similar appearance.

The remainder of this chapter is organised as follows. In Section 4.2 we discuss some work related to the content of this chapter. Following that in Section 4.3 we present our novel clustering algorithm. In Section 4.4 we evaluate the method in different experiments before concluding with a summary of the chapter's content in Section 4.5

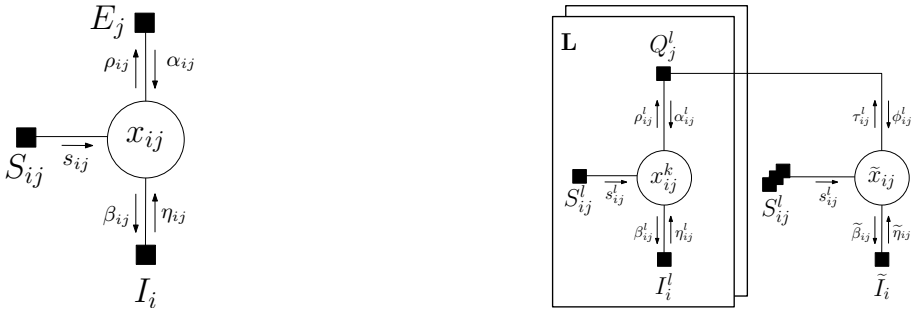
The work presented in this chapter was previously published in IROS 2013 (Ott and Ramos, 2013a).

4.2 Related work

In machine learning, there have been several extensions to affinity propagation addressing hierarchical clustering. Xiao et al. (2007) propose a greedy hierarchical model in which each subsequent layer is based on the exemplars of the previous layer. The method proposed by Givoni et al. (2011) uses a graphical model which connects subsequent layers, showing it can outperform the simple greedy approach. A two-layer hierarchical model is proposed by Wang et al. (2013). Their method jointly finds exemplars and associated clusters. These last two methods derive a set of update rules from a graphical model. In contrast to these our work uses a lateral rather then a hierarchical model. It is, however, also based on a graphical model from which we derive message updates.

There are also other methods designed to cluster data from multiple sources. Zhang et al. (2004) propose a Markov random field model with mutual information as potential functions to cluster data with multiple modalities. Another method, based on spectral clustering, was proposed by Bekkerman and Jeon (2007) in which a k-partite graph based on the input data is built. This graph is then used to derive the matrices needed by spectral clustering.

In another line of research, the combination of different sensor modalities has been shown to improve perception performance. Triebel et al. (2012) process point cloud data into a mesh and compute features to segment scenes and identify objects in the scene. Jebari and Filliat (2012) performed object segmentation by combining depth and colour features of superpixels with a Markov random field. Schoenberg et al. (2010) presented another method employing a Markov random field to cluster RGB-D data in order to segment urban scenes. In Howard et al. (2006) texture and geometric features are used to learn terrain type and traversability from stereo camera data. The clustering of these features is per-



(a) AP Messages (b) LAP Messages

Figure 4.1: The messaging structure for (a) affinity propagation and (b) layered affinity propagation. The Q factor node is in a sense an augmented version of the E factor node of the original affinity propagation.

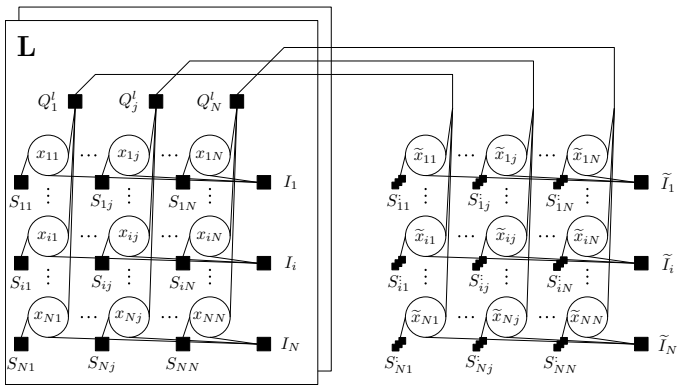


Figure 4.2: The complete factor graph used in layered affinity propagation. The left side shows the L data layers while the right side shows the merging layer.

formed using a distance function that weights the different features. A different approach to learning terrain traversability is taken by Sun et al. (2005) who use shape and colour as features. Those are then clustered using an ad-hoc clustering method based on the feature similarity. Katz et al. (2010) use a linear weighted combination of visual and laser stamps to detect dynamic obstacles in the environment. All of these methods require the user to manually define how the different features are to be combined. Our method, in contrast, only requires the definition of features for each data source separately. How these are combined is a task automatically solved by the algorithm.

4.3 Layered affinity propagation

The idea behind our algorithm is to allow the data of each sensor to be clustered individually, in so called data layers, while imposing the constraint that the solution of each of these layers is identical. During optimisation this will lead to trade-offs where some layers have to choose suboptimal solutions to improve the

overall solution. The cost function we wish to optimise is an extension of the one used by affinity propagation and has the following form:

$$C(\{x_{ij}^{\cdot}, \widetilde{x}_{ij}\}) = \sum_{i,j,l} S_{ij}^{l}(x_{ij}^{l}) + \sum_{i,l} I_{i}^{l}(x_{i:}^{l}) + \sum_{i} \widetilde{I}_{i}(\widetilde{x}_{i:}) + \sum_{j,l} Q_{j}^{l}(x_{:j}^{l}, \widetilde{x}_{:j}) \,, \tag{4.1}$$

where x_{ij}^{l} is the binary assignment variable between point i and j in data layer l, $\widetilde{x}_{ij}$ is the binary variable between point i and j in the merging layer. The merging layer is responsible for enforcing that the data layers agree on a common solution. The different terms of the energy function are defined as follows:

$$S_{ij}^{l} = \begin{cases} s_{ij}^{l} & \text{if} x_{ij}^{l} = 1 \\ 0 & \text{otherwise} \end{cases} \tag{4.2}$$

$$I_{i}^{l}(x_{i:}^{l}) = \begin{cases} 0 & \text{if } \sum_{j} x_{ij}^{l} = 1 \\ -\infty & \text{otherwise} \end{cases} \tag{4.3}$$

$$\widetilde{I}_{i}(\widetilde{x}_{i:}) = \begin{cases} 0 & \text{if } \sum_{j} \widetilde{x}_{ij} = 1 \\ -\infty & \text{otherwise} \end{cases} \tag{4.4}$$

$$Q_{j}^{l}(x_{:j}^{l}, \widetilde{x}_{:j}) = \begin{cases} 0 & \text{if } x_{jj}^{l} = \max_{i} x_{ij}^{l} \wedge x_{jj}^{l} = x_{jj}^{k} \forall l, k \\ -\infty & \text{otherwise} \end{cases} . \tag{4.5}$$

These constraints can be interpreted as follows. I_{i}^{l} and $\widetilde{I}_{i}$ ensure that each point is assigned to only one exemplar in the data and merging layer respectively. Q_{j}^{l} connects the individual data layers with the merging layer and ensures that (i) only valid exemplars are selected and (ii) that the same exemplars are chosen by all layers. We can represent the cost function as a factor graph, as shown in Figure 4.2. The left part of the graphical model shows the data layers which do not communicate with each other directly. Each of these data layers is connected to the merging layer, shown on the right side of the model. The different messages exchanged in this model are displayed in Figure 4.1b, the original affinity propagation messages are shown in Figure 4.1a for comparison.

Comparing the messages exchanged by affinity propagation and layered affinity propagation we can see that the data layers L are very similar to standard affinity propagation. Both methods have a factor node I which ensures that every data point is assigned to exactly one cluster. The difference comes from the Q factor node, which replaces the E factor node. This factor enables communication between the data layers and the merging layer. The role though stays the same with the addition that the exemplar consistency constraint is enforced over the entire network. The merging layer has a few more differences as it merges the informa-

tion from all data layers through the Q nodes into its own cluster assignments. It also accesses the similarities $S^{:}_{ij}$ of all data layers. While the model allows the values of $S^{:}_{ij}$ to differ between the data layer and the merging layer we have kept the values identical. The merging layer uses the information from the data layers to come up with its own decision which is propagated back to the data layers and thus information is shared between all layers. All this is encoded in the energy function Eq. (4.1) which we optimise again by finding its MAP assignments.

The update messages are derived in the same way they are derived for affinity propagation. It involves the computation of the difference in settings as well as simplifications based on the constraints. This results in the following messages for the data layers:

$$s^l_{ij} = S^l_{ij} \tag{4.6}$$
$$\beta^l_{ij} = s^l_{ij} + \alpha^l_{ij} \tag{4.7}$$
$$\eta^l_{ij} = -\max_{k \neq j} \beta^l_{ik} \tag{4.8}$$
$$\rho^l_{ij} = s^l_{ij} + \eta^l_{ij} \tag{4.9}$$
$$\alpha^l_{ij} = \min\left[0, \rho^l_{jj} + \tau^l_{jj} + \sum_{k \notin \{i,j\}} \max(0, \rho^l_{kj}) + \sum_{k \neq j} \max(0, \tau^l_{kj})\right] \tag{4.10}$$
$$\alpha^l_{jj} = \tau^l_{jj} + \sum_{k \neq j} \max(0, \rho^l_{kj}) + \sum_{k \neq j} \max(0, \tau^l_{kj}) \ . \tag{4.11}$$

While the messages of the merging layer have the following form:

$$\widetilde{\beta}_{ij} = \sum_t s^t_{ij} + \sum_t \phi^t_{ij} \tag{4.12}$$
$$\widetilde{\eta}_{ij} = -\max_{k \neq j} \widetilde{\beta}_{ik} \tag{4.13}$$
$$\tau^l_{ij} = \widetilde{\eta}_{ij} + \sum_t s^t_{ij} + \sum_{t \neq l} \phi^t_{ij} \tag{4.14}$$
$$\phi^l_{ij} = \min\left[0, \tau^l_{jj} + \rho^l_{jj} + \sum_{k \notin \{i,j\}} \max(0, \tau^l_{kj}) + \sum_{k \neq j} \max(0, \rho^l_{kj})\right] \tag{4.15}$$
$$\phi^l_{jj} = \rho^l_{jj} + \sum_{k \neq j} \max(0, \tau^l_{kj}) + \sum_{k \neq j} \max(0, \rho^l_{kj}). \tag{4.16}$$

These messages can be further simplified through substitution, yielding:

$$\begin{aligned} \rho^l_{ij} &= s^l_{ij} + \eta^l_{ij} \\ &= s^l_{ij} - \max_{k \neq j} \beta^l_{ik} \\ &= s_{ij} - \max_{k \neq j} \left(s^l_{ik} + \alpha^l_{ik}\right) \end{aligned} \tag{4.17}$$

Algorithm 10: Layered affinity propagation

Input: Similarity matrices S^l
Output: Exemplars e and assignments a

$\rho_{ij}^l, \tau_{ij}^l, \alpha_{ij}^l, \phi_{ij}^l \leftarrow 0$
repeat
  for $l \in L$ **do**
    $\rho_{ij}^l \leftarrow s_{ij} - \max_{k \neq j} \left(s_{ik}^l + \alpha_{ik}^l \right)$
    $\alpha_{ij}^l \leftarrow \min \left[0, \rho_{jj}^l + \tau_{jj}^l + \sum_{k \notin \{i,j\}} \max(0, \rho_{kj}^l) + \sum_{k \neq j} \max(0, \tau_{kj}^l) \right]$
  end
  for $l \in L$ **do**
    $\tau_{ij}^l \leftarrow \sum_t s_{ij}^t - \sum_{t \neq l} \phi_{ij}^t + \max_{k \neq j} \left(\sum_t s_{ij}^t + \sum_t \phi_{ij}^t \right)$
    $\phi_{jj}^l \leftarrow \rho_{jj}^l + \sum_{k \neq j} \max(0, \tau_{kj}^l) + \sum_{k \neq j} \max(0, \rho_{kj}^l)$
  end
until *convergence*
$e \leftarrow$ points for which $\sum_l (\tau_{jj}^l + \phi_{jj}^l) > 0$ holds
$a_i \leftarrow$ assign point i to exemplar e satisfying $\operatorname{argmax}_e \sum_l (\tau_{ie}^l + \phi_{ie}^l)$
return e, a

$$\begin{aligned} \tau_{ij}^l &= \sum_t s_{ij}^t + \sum_{t \neq l} \phi_{ij}^t + \widetilde{\eta}_{ij} \\ &= \sum_t s_{ij}^t - \sum_{t \neq l} \phi_{ij}^t + \max_{k \neq j} \left(\widetilde{\beta}_{ik} \right) \\ &= \sum_t s_{ij}^t - \sum_{t \neq l} \phi_{ij}^t + \max_{k \neq j} \left(\sum_t s_{ij}^t + \sum_t \phi_{ij}^t \right). \end{aligned} \tag{4.18}$$

The pseudo code of the algorithm is shown in Algorithm 10. First, all messages are initialised to 0. Next, we update the local data layer messages ρ_{ij}^l and α_{ij}^l before merging the results by computing τ_{ij}^l and ϕ_{ij}^l. These two steps are performed until we achieve convergence. Convergence is achieved when the net similarity stays stable over a few iterations. The final solution is extracted by first finding the exemplars as the points for which $\sum_l \left(\tau_{jj}^l + \phi_{jj}^l \right) > 0$. The remaining points are then assigned to the exemplar e which satisfies $\operatorname{argmax}_e \sum_l \left(\tau_{ie}^l + \phi_{ie}^l \right)$.

The algorithm requires $O(5LN^2)$ storage for the messages and similarities. The runtime is $O((L+1)N^2)$ per iteration where L is the number of data layers and N is the number of data points, as we need to run affinity propagation for every layer and the merging layer. Note that the L data layers can all be run in parallel as they do not influence each other directly. With the ubiquity of multi-core CPUs distributing the computation of message updates over multiple cores should allow for significant reductions in runtime.

4.4 Experiments

In this section we evaluate the proposed method in two different applications. First we demonstrate that layered affinity propagation (LAP) can be used to perform scene segmentation on Kinect data. Next, we cluster data obtained from a Velodyne and camera pair. We compare our method against affinity propagation using only colour or depth information as well as k-means using a combined colour and depth feature vector. In all experiments the self-similarity values S_{jj} were set to $\theta \cdot median(S)$, $\theta \in [2, 10]$. In practice this results in an adequate number of clusters without favouring any single point in the exemplar choice. Convergence of affinity propagation is achieved once the similarity score of the assignments is stable over a number of iterations, 20 in our case. In the experiments convergence was never an issue and typically achieved after 100 to 200 iterations.

In both experiments the ground truth required for the evaluation using V-Measure was obtained by manually labelling the data. The labels represent the class of an observation rather then individual instances, as we are interested in the ability to group the data into larger concepts as opposed to recognizing specific objects.

4.4.1 Indoor RGB-D segmentation

In this experiment we evaluate how well combining different features using LAP performs at segmenting scenes captured with a Kinect. The Kinect provides us with dense depth and colour information with a 1:1 mapping between depth and colour pixels. We start by over-segmenting the data by extracting super pixels from the image using SLIC (Achanta et al., 2012). From these super pixels we extract colour and depth features which we subsequently cluster to obtain the final segmentation. In this experiment we use LAB colour histograms and average surface normals as our features. The similarity values required by affinity propagation are computed using the Bhattacharyya distance for colour histograms and angular difference between vectors for the mean surface normals. The features and similarity measures used are summarised in Table 4.2.

k-means operates directly on the two histograms and is set to find 10 clusters. For a more direct comparison to k-means we additionally run affinity propagation with a similarity matrix obtained from the Euclidean distance between the feature histograms. For both k-means and affinity propagation we choose reasonable parameters but no search for the optimal parameter set is performed, as this would not demonstrate the typical performance of the methods. To see how clustering methods perform in comparison to dedicated segmentation approaches we also use colour and smoothness based region growing methods.

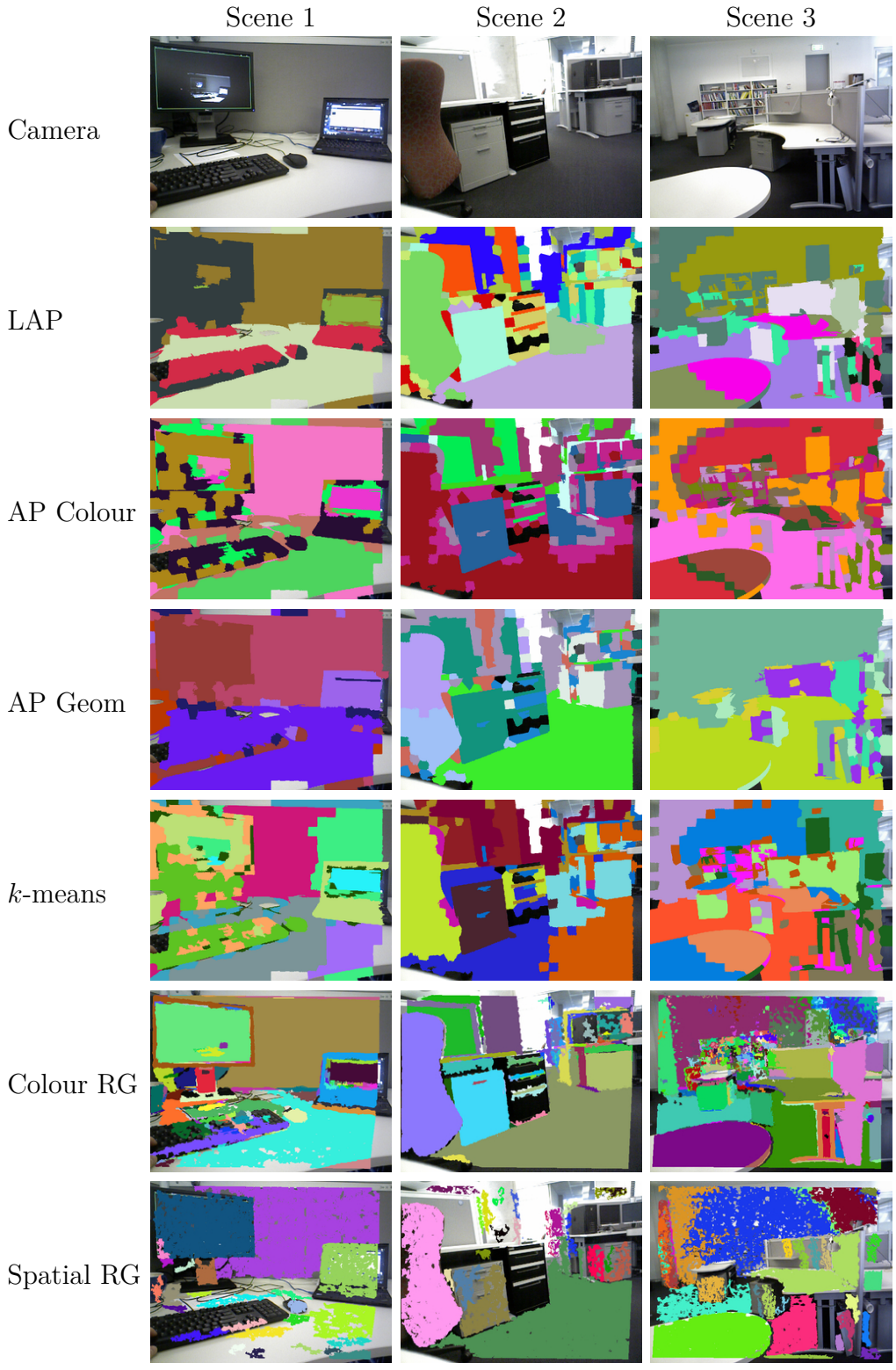


Figure 4.3: Exemplary segmentation results for three different scenes. From top to bottom we have: raw image, LAP, colour affinity propagation, geometry affinity propagation, k-means, colour based region growing and spatial region growing. The colours indicate the cluster assignments made.

Method	V-Measure	Homogeneity	Completeness
	Scene 1		
LAP	0.48	0.49	0.47
Colour AP	0.32	0.36	0.29
Geometry AP	0.52	0.53	0.51
Combined AP	0.41	0.51	0.34
k-means	0.36	0.46	0.29
	Scene 2		
LAP	0.56	0.55	0.57
Colour AP	0.40	0.36	0.46
Geometry AP	0.45	0.41	0.51
Combined AP	0.54	0.57	0.52
k-means	0.52	0.50	0.54
	Scene 3		
LAP	0.44	0.45	0.44
Colour AP	0.36	0.36	0.36
Geometry AP	0.36	0.30	0.44
Combined AP	0.45	0.47	0.44
k-means	0.45	0.43	0.48
	Overall		
LAP	0.48 ± 0.09	0.49 ± 0.10	0.47 ± 0.09
Colour AP	0.40 ± 0.09	0.40 ± 0.10	0.40 ± 0.10
Geometry AP	0.43 ± 0.13	0.41 ± 0.14	0.45 ± 0.14
Combined AP	0.43 ± 0.10	0.46 ± 0.12	0.40 ± 0.12
k-means	0.41 ± 0.06	0.48 ± 0.10	0.37 ± 0.08

Table 4.1: V-Measure, homogeneity and completeness scores for the four methods evaluated for the three scenes shown in Figure 4.3 as well as all the recorded scenes.

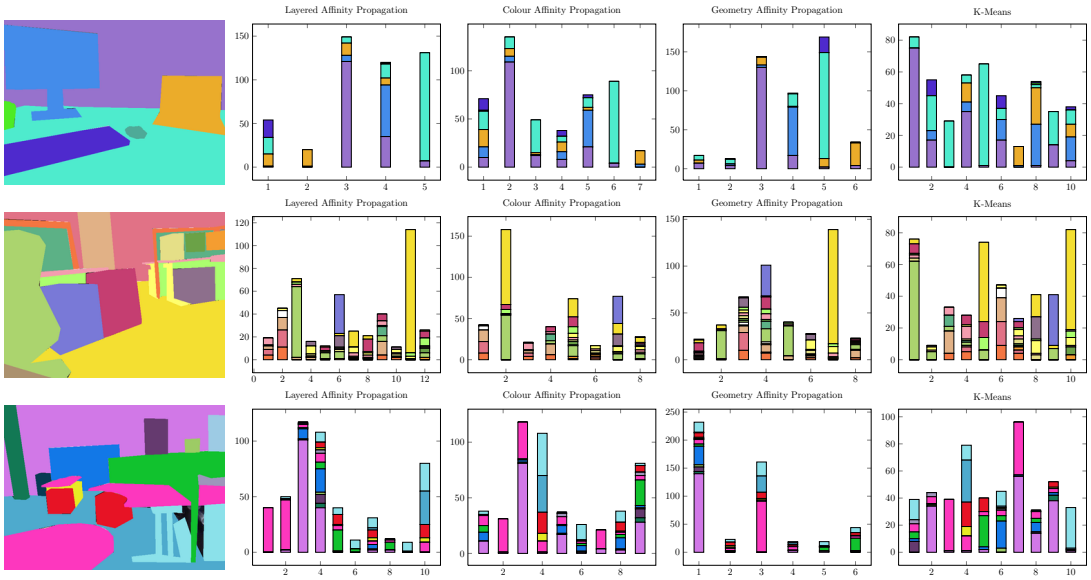


Figure 4.4: Clustering statistics for the results shown in Figure 4.3. The first column contains ground truth labels for each of the scenes while the four plots to the right show the ground truth label distribution in the clusters for each of the four methods used. Each bar represents a single cluster and visualises the number of data points in the cluster via its height. Homogeneity of a cluster is visualised by the number of different colours in the bar. Completeness can be assessed by the distribution of a single colour over all clusters

We collected several scenes in a typical office environment containing chairs, books, binders, desks, shelves, computers, etc. Typical results for both clustering and region growing methods are shown in Figure 4.3. Each row contains the results obtained with the method shown in the first column. The different colours in the images indicate the segmentation produced by the method.

One big difference between clustering and region growing methods is that typically clustering methods do not consider spatial closeness and thus may group spatially distant but similar objects together. This can be seen in the second scene with the grey drawers or table surfaces in the third scene. Whether or not

Data Type	Feature	Similarity Measure
	Indoor RGB-D	
Colour image	LAB histograms	Bhattacharyya distance
Pointcloud	Surface normals	Angular vector difference
	KITTI	
Colour image	LAB histograms	Bhattacharyya distance
Pointcloud	Normal histograms	Bhattacharyya distance

Table 4.2: Features and similarities used with the indoor RGB-D and KITTI datasets.

this is a desirable property depends on the application. However, adding spatial connectivity information into the clustering system would allow it to exhibit a more region growing like behaviour. Adapting a region growing approach to behave more like clustering methods though is not possible.

Figure 4.3 shows quantitative evaluation of results obtained with the different clustering methods in three scenes. Looking at colour AP and geometry AP it is obvious that the clusters they find correspond to the features used. However, this is problematic as for example the depth feature is unable to distinguish between the surface of a table and the floor. From the k-means results we see how having both modalities improves the results. However, the choice of the number of clusters can have a big impact on the result as it can lead to under or over segmentation. If we now look at results obtained with LAP we can see that clusters adhere well to object boundaries with most larger areas being successfully clustered as a single region when compared to k-means which often ends up splitting them. The numerical evaluation using V-Measure (Rosenberg and Hirschberg, 2007) in Table 4.1 shows a general trend where LAP outperforms k-means while colour AP and geometry AP come in last. Comparing k-means to Combined AP, which uses the Euclidean distance metric on the feature vectors used by k-means, we can see how AP outperforms k-means even when using the same metric. However, LAP is still able to improve on these results indicating that a more principled way of combining the data is beneficial. Looking at the individual results of the three scenes we can see that geometry AP performs better when the scene is composed of a few large and distinct areas, as is the case in scene one. This is not always beneficial though, as we see in scene 3, where the book shelve is better segmented by colour AP while geometry AP simply merges it into the wall. These points show how useful it can be to merge multiple different sensor modalities which is also reflected by the scores obtained by LAP.

For a more detailed analysis of the results we visualise the size, homogeneity and completeness of each cluster in Figure 4.4. Each bar represents a single cluster with the distribution of true labels in it, given by the labelled image to the left. The height indicates the cluster's size while the distribution of true labels within a bar represents its homogeneity. A cluster's completeness can be assessed by the distribution of a single label over all bars. These plots show how geometry AP tends to form a few large clusters which capture the major surface normals in the environment which explains the tendency to under segment scenes. For colour AP we can observe how most clusters contain multiple different labels such as the second cluster in the second scene which contains both the chair and floor. This shows how colour AP fails to separate areas that appear similar based on colour histograms. The k-means results exhibit a more uniform size then the

Figure 4.5: Visualisation of exemplary point clouds of the KITTI dataset coloured using information from the camera images.

other methods with a mixture of very homogeneous clusters and mixed clusters. Finally, LAP produces uniform clusters for large areas in the scenes and at times collapses multiple small classes into a single cluster.

Observing the results of the two region growing methods we see that they produce cleaner results compared to the clustering methods. Still, each have their own set of drawbacks. The colour based version is prone to oversegmentation if there is no direct connectivity between components which is easily caused by occlusions. The smoothness based version has the same need for connectivity but is much more sensitive to a good smoothness threshold choice. A different set of thresholds could potentially provide better results for the wall in scene three though this would cause other parts to be undersegmented.

4.4.2 KITTI dataset

The KITTI dataset (Geiger et al., 2012) is designed as a vision benchmark but also provides calibrated Velodyne and camera data recorded in urban environments. This provides us with the same data modalities as the Kinect, depth and colour, but at different densities and no direct correspondence between the two modalities. Figure 4.5 shows the type of coloured point clouds this dataset provides us with. In this experiment we are not clustering raw point clouds but rather segments extracted from these. To this end we segment process the data as follows:

1. remove the ground plane from the point cloud;
2. find segments in the point cloud using Euclidean distance clustering;
3. extract image parts corresponding to the point cloud segment.

Figure 4.6: Exemplary image data corresponding to point cloud segments extracted from the raw Velodyne data. Shown are cars, pedestrians, cyclists and wall segments.

This provides us with 3D point clouds with associated colour information. From this data we select segments which occur frequently, i.e. cars, cyclists, pedestrians and wall segments. Examples of such segments are depicted in Figure 4.6 which shows the variability the data has in orientation, posture, colour, and size, both within and between classes. From this collection of segments we choose random subsets to cluster. The features we extract, listed in Table 4.2, are colour histograms and surface normal histograms. Bhattacharyya distance is used to compute pairwise similarities. We show the average results obtained from 20 runs in Table 4.3. We can see how combining the two modalities with LAP improves the results. The colour and geometry based affinity propagation methods produce decent results but are outperformed by LAP. k-means on the other hand struggles on this data set. Adjusting the number of clusters could improve the result somewhat but no single value would work for all runs. This reinforces the importance of methods that select the appropriate number of clusters automatically. Another important observation is the high homogeneity score of LAP which means that using hierarchical methods can easily further improve the results. High homogeneity indicates that most clusters consist of a single type of object and therefore merging them with an hierarchical method will produce clusters that are more complete without reducing the homogeneity.

To better understand the results we show segments successfully assigned to the same cluster by LAP in Figure 4.7. This shows how cars, pedestrians and cyclists are grouped together even though they appear different in colour and posture. While at first this may be counter intuitive we have to remember that affinity propagation optimises a global cost function and as such is influenced by within cluster similarity as well as inter cluster dissimilarity. For example cars tend to use a single colour and have one or two strong normals, whereas a pedestrian will have

Figure 4.7: Examples of successful clustering results obtained with LAP. The individual groups show cars, pedestrians and cyclists respectively clustered together despite their different appearances.

Figure 4.8: This shows one of the more common clustering mistakes, the side view of a car being clustered together with wall segments. Since only the planar side of the van is visible and both walls and van have rather uniform colour distributions this type of error is not surprising.

multiple main colours and more evenly distributed normals. Thus by optimising both the similarity within a cluster as well as the dissimilarity between clusters the algorithm is capable of finding the solutions shown. In Figure 4.8 we show an instance where the clustering failed to separate a van from wall segments. Since only the side of the van is visible it is easy to see why these segments were grouped together. While we ideally would like the clustering to provide us with four clusters representing our four classes it is unrealistic to achieve this directly. However, the homogeneous nature of LAP clusters should allow us to use hierarchical methods to improve the results.

4.4.3 Convergence behaviour

To investigate the impact on convergence of the more complicated graphical model of LAP compared to that of standard AP we compare the evolution of the solution energy as the number of iterations increases. Figure 4.9 shows the energy for both LAP and AP on data from the Kinect experiment. Looking at the results for LAP

Method	V-Measure	Homogeneity	Completeness
LAP	0.41 ± 0.01	0.82 ± 0.02	0.28 ± 0.01
Colour AP	0.35 ± 0.01	0.66 ± 0.02	0.26 ± 0.01
Geometry AP	0.37 ± 0.01	0.67 ± 0.02	0.26 ± 0.01
k-means	0.14 ± 0.02	0.19 ± 0.02	0.11 ± 0.01

Table 4.3: Average V-Measure, homogeneity and completeness scores with standard deviation of 20 clustering runs. LAP has the best overall V-Measure score but also produces much more homogeneous results. This is important as it indicates that further hierarchical processing is likely to improve the results.

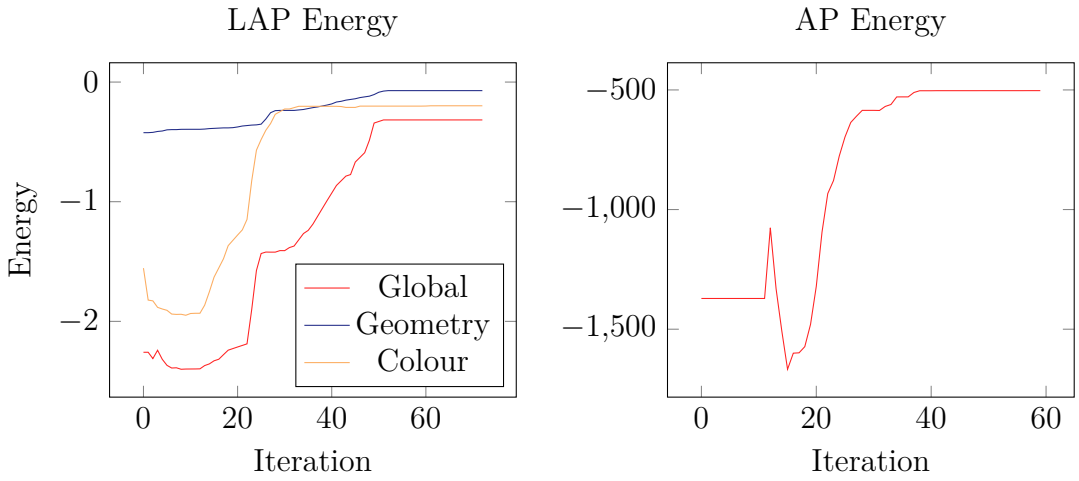


Figure 4.9: This plots shows the energy of the solution over iterations of both LAP (left) and AP (right) on data from the Kinect experiment. In the case of LAP the energies for the merging layer as well as the two data layers for colour and geometry data. While the energy values of the two methods can not be compared, both LAP and AP exhibit similar curves and converge to a solution in a reasonable amount of iterations.

we can see that the two modalities, colour and geometry, have different levels of improvements. While colour improves quite drastically the geometry based layer improvements are less pronounced. However, the two data layers and the global merging layer all converge together to a stable solution in about 70 iterations. Comparing this to AP, which uses a concatenated feature vector of colour and geometry data, we can see that it requires around 60 iterations but has a similar convergence pattern otherwise. This example of typical behaviour shows that LAP tends to converge in a similar number of iterations as standard AP does with all parts of the model converging. While the energy values are not directly comparable the overall shapes of the curves can be compared.

4.5 Summary

In this chapter we presented a novel clustering approach that combines information from multiple sensors in a principled way. The approach enables the user to define features that are appropriate for each sensor individually, without having to worry about how to best combine them. In experiments we have shown that this approach can be used to perform segmentation by combining colour and depth information of a Kinect. In a second experiment we evaluated the performance of the method when clustering point cloud segments with colour information obtained in urban scenes. While the experiments concentrated on depth and colour information nothing prevents the use of data from other sensors, such as accelerometers or hyperspectral cameras for example.

Chapter 5

Joint clustering and outlier detection

5.1 Introduction

The data provided by sensors in robotics always contains noise and outliers which can stem from invalid observations, malfunctioning sensors or other unforeseen events. Typically, the goal is to remove such observations during processing as they can cause problems. A commonly used approach to this end is RANSAC (RANdom SAmple Consensus) (Fischler and Bolles, 1981), which attempts to fit a model to a subset of the dataset which contains outliers. For example, in visual odometry matches between frames are typically subjected to RANSAC filtering to remove matches that do not fit the assumed model.

Outliers, however, can also indicate a novel process and as such can provide important new information. For long-term autonomy it is expected that changes in the environment are initially identified as outliers. The methods presented so far did not attempt to handle outliers separately. Here we propose a novel method that handles clustering and outlier selection in a unified manner. The outputs of the method are both the clusters as well as the outliers found in the data. This is important as not only can outliers be useful later on but also improve the robustness of the clustering method.

In this chapter we present an integer program formulation for joint clustering and outlier selection. Two algorithms are proposed to solve this optimisation task. The resulting algorithms require distance measurements between pairs of points as well as the number of outliers ℓ to be found. The methods proposed are: (i) an extension to affinity propagation (Frey and Dueck, 2007) for clustering and outlier detection (APOC) and (ii) a Lagrangian duality relaxation algorithm to solve the same optimisation problem (LR). The main contributions of this chapter are:

- formulation of the clustering with outlier selection problem as an integer program;
- extension of affinity propagation to solve the integer program;
- a scalable algorithm based on the Lagrangian duality;

- evaluation on synthetic and real-world data sets.

The remainder of this chapter is structured as follows. In Section 5.2 we present related work before we present the general problem and two algorithms for solving the problem in Section 5.3. In Section 5.4 we evaluate the methods on both synthetic and real datasets. Finally, Section 5.5 concludes this chapter with a short summary.

The work presented in this chapter was previously published in NIPS 2014 (Ott et al., 2014).

5.2 Related work

Clustering and outlier detection are often studied as two separate problems (Chandola et al., 2009). However, it is natural to consider them simultaneously, as outliers can have a disproportionate impact on the location and shape of clusters, which in turn can help to identify, contextualise and interpret outliers.

The field of robust statistics studies the design of statistical methods which are less sensitive to the presence of outliers (Huber and Ronchetti, 2008). For example, the median and trimmed mean estimators are far less sensitive to outliers than the mean. Similarly, versions of principal component analysis (PCA) have been proposed (Croux and Ruiz-Gazen, 1996; Wright et al., 2009) which are more robust against model misspecification. An important primitive in the area of robust statistics is the notion of minimum covariance determinant (MCD) (Rousseeuw, 1984). The objective is to identify a subset in a multivariate dataset of n points that minimises the determinant of the variance-covariance matrix over all subsets of size $n - \ell$, given a user defined value of ℓ. The resulting variance-covariance matrix can be integrated into the Mahalanobis distance and used as part of a chi-square test to identify multivariate outliers (Rousseeuw and Driessen, 1999).

In theoretical computer science similar problems have been studied in the context of clustering and facility location. For example, Chen (2008) proposes a constant factor approximation algorithm for the k-median with outliers problem. The idea is to minimise the cost of k-median clustering by removing ℓ points from a dataset with n points. Charikar et al. (2001) have proposed a bi-criteria approximation algorithm for the facility location with outliers problem. While of theoretical interest, none of these algorithms are amenable to a practical implementation on large data sets.

Robustness of clustering methods in general is discussed in (Garcia-Escudero et al., 2010; Hennig, 2008). They provide a good theoretical overview about the

conditions under which clustering methods can deal with noise or outliers. However, it is difficult to determine "a priori" if data exhibits the required properties.

More recently, Chawla and Gionis (2013) have proposed k-means-- a practical and scalable algorithm for the k-means with outlier problem. The method is a two step approach where in each iteration of standard k-means, the ℓ points with the largest distance to their centroid are ignored in the next round of k-means. However, the algorithm inherits the weaknesses of classical k-means, i.e., (i) the requirement of setting the number of clusters k and (ii) initial selection of the k centroids. The trimmed k-means (Cuesta-Albertos et al., 1997) algorithm is a special case of k-means-- with $k = 1$.

In data mining Knorr and Ng (1997) proposed a definition of distance-based outliers which relaxed strict distributional assumptions and was readily generalisable to multi-dimensional data sets. Following Knorr and Ng, several variations and algorithms have been proposed to detect distance-based outliers (Bay and Schwabacher, 2003; Ramaswamy et al., 2000). However, the outliers detected by these methods are global outliers, i.e., the "outlierness" is with respect to the whole dataset. Breunig et al. (2000) have argued that in some situations local outliers are more important than global outliers and cannot be easily detected by standard distance-based techniques. They introduced the concept of local outlier factor (LOF) which captures how isolated an object is with respect to its surrounding neighbourhood. The concept of local outliers has subsequently been extended in several directions (Chandola et al., 2009; Chawla and Sun, 2006; Papadimitriou et al., 2003).

In robotics, outliers are often considered a nuisance and methods typically attempt to filter them out with methods such as RANSAC. For example, in visual odometry the matching process is made robust by using RANSAC to remove wrong matches (Nistér et al., 2004). To obtain a robust place recognition method Olson (2009) uses Single Cluster Graph Partitioning (SCGP) (Olson et al., 2005), as an alternative to RANSAC, to reject outliers in local matches to create a single high quality global match. Paul and Newman (2011) proposed a system based on latent Dirichlet allocation (LDA) to identify the most surprising images in order to improve the model for these specific images. Girdhar et al. (2014b) use a realtime online spatiotemporal topic modelling technique (Girdhar et al., 2014a) to build a terrain representation based on the robot's observations. This model is then used to plan paths of high information gain based on metrics such as word and topic perplexity. In both cases these perplexing observations can be considered outliers in the observed dataset as the model struggles to explain them.

5.3 Optimisation formulation

5.3.1 Integer program

Given an assignment cost matrix d_{ij}, cluster creation costs c_j, and the number of outliers ℓ we define the task of clustering and outlier selection as the problem of finding the assignments to the binary exemplar indicators y_i, outlier indicators o_i and point assignments x_{ij} that minimises the following cost function:

$$\begin{aligned}
\text{minimise} \quad & \sum_j c_j y_j + \sum_i \sum_j d_{ij} x_{ij} && \text{(5.1a)} \\
\text{subject to} \quad & x_{ij} \leq y_j && \text{(5.1b)} \\
& o_i + \sum_j x_{ij} = 1 && \text{(5.1c)} \\
& \sum_i o_i = \ell && \text{(5.1d)} \\
& x_{ij}, y_j, o_i \in \{0, 1\}. && \text{(5.1e)}
\end{aligned}$$

The above constraints enforce valid solutions, i.e.:

- points can only be assigned to valid exemplars (Eq. (5.1b));
- every point must be assigned to exactly one other point or be declared an outlier (Eq. (5.1c));
- exactly ℓ outliers have to be selected (Eq. (5.1d));
- only integer solutions are allowed (Eq. (5.1e)).

This formulation describes the facility location problem (Hamacher and Drezner, 2002) with the addition of outlier selection (FLO). Optimising this cost function automatically selects the optimal number of clusters as well as the most likely outliers. This formulation can be directly solved using linear programming packages such as CPLEX. We will refer to this solution as FLO_{LP}, the optimal LP solution.

In the following we present two different methods to solve this optimisation problem. The first method is an extension to affinity propagation while the second one exploits the Lagrangian duality formulation of the original problem.

5.3.2 Affinity propagation outlier clustering (APOC)

The extension to affinity propagation, based on the binary variable model (Givoni and Frey, 2009) solves the integer program of Section 5.3.1 by representing it with the factor graph shown in Figure 5.1a. The factor graph consists of two parts.

The left side is identical to AP, with the difference that it is connected to the right side responsible for the selection of outliers. The constraints of the integer program Eq. (5.1a) are encoded by the following energy function:

$$\max \sum_{ij} S_{ij}(x_{ij}) + \sum_{j} E_j(x_{:j}) + \sum_{i} I_i(x_{i:}, o_{i:}) + \sum_{k} P_k(o_{:k}), \tag{5.2}$$

where

$$S_{ij}(x_{ij}) = \begin{cases} -c_i & \text{if } i = j \\ -d_{ij} & \text{otherwise} \end{cases} \tag{5.3}$$

$$I_i(x_{i:}, o_{i:}) = \begin{cases} 0 & \text{if } \sum_j x_{ij} + \sum_k o_{ik} = 1 \\ -\infty & \text{otherwise} \end{cases} \tag{5.4}$$

$$E_j(x_{:j}) = \begin{cases} 0 & \text{if } x_{jj} = \max_i c_{ij} \\ -\infty & \text{otherwise} \end{cases} \tag{5.5}$$

$$P_k(o_{:k}) = \begin{cases} 0 & \text{if } \sum_i o_{ik} = 1 \\ -\infty & \text{otherwise} \end{cases} \tag{5.6}$$

with $x_{i:} = x_{i1}, \ldots, x_{iN}$. Due to the nature of affinity propagation we maximise the energy function and use negative distances. The three constraints can be interpreted as follows:

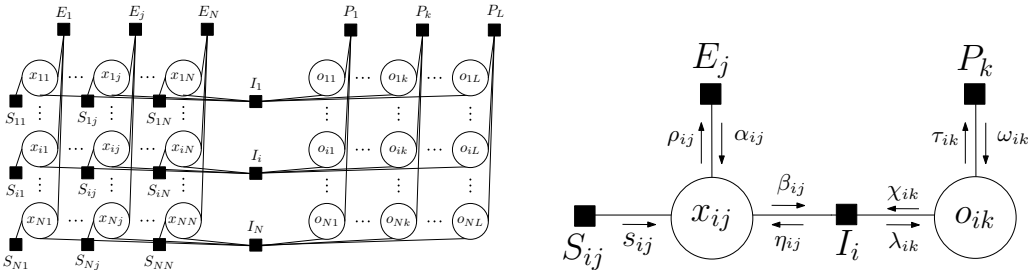
1. 1-of-N Constraint (I_i). Each data point has to choose exactly one exemplar or be declared as an outlier (Eq. (5.1c)).

2. Exemplar Consistency Constraint (E_j). For point i to select point j as its exemplar, point j must declare itself an exemplar (Eq. (5.1b)).

3. Select ℓ Outliers Constraint (P_k). For every outlier selection exactly one point is assigned (Eq. (5.1d)).

These constraints are enforced by associating an infinite cost with invalid configurations, resulting in an obviously suboptimal solution. As with all affinity propagation based methods we find the maximum a posteriori settings (MAP) using the max-sum algorithm (Kschischang et al., 2001). The message update rules required for this are obtained by using the above constraints to simplify the general update messages (Equations (2.13) and (2.14)).

The messages exchanged by APOC are shown in Figure 5.1b. We can see that each node x_{ij} is connected to three factors: S_{ij}, I_i and E_j, whereas outlier nodes o_{ik} are connected to only two, I_i and P_k. Messages ρ_{ij}, β_{ij}, τ_{ik} and χ_{ik} are sent from nodes to factors and derived using Eq. (2.13). The other five messages



(a) Factor graph (b) Messages

Figure 5.1: (a) Graphical model of APOC, the left part is responsible for the clustering of the data, while the right part is responsible for the outlier selection. These two parts interact with each other via the I factor nodes. (b) Messages exchanged by APOC, x_{ij} represents the clustering choice whereas o_{ik} represents the outlier choice.

$s_{ij}, \alpha_{ij}, \eta_{ij}, \lambda_{ik}$ and ω_{ik} are derived with Eq. (2.14) since they are sent from a factor to a node. Since only binary variables are involved, it is sufficient to compute the difference between the two variable settings. Combining and simplifying these messages we obtain the final set of update equations as:

$$\rho_{ij} = s_{ij} + \min\left[-\max_{t\neq j}(\alpha_{it} + s_{it}), -\max_{t}(\omega_{it})\right], \tag{5.7}$$

$$\alpha_{ij} = \begin{cases} \sum_{t\neq j}\max(0,\rho_{tj}) & i = j \\ \min\left[0, \rho_{jj} + \sum_{t\notin\{i,j\}}\max(0,\rho_{tj})\right] & i \neq j \end{cases}, \tag{5.8}$$

$$\lambda_{ik} = \min\left[-\max_{t}(\alpha_{it} + s_{it}), -\max_{t\neq k}(\omega_{ti})\right], \tag{5.9}$$

$$\omega_{ik} = -\max_{t\neq i}(\lambda_{tk}). \tag{5.10}$$

With the messages defined we can now outline the complete algorithm, as shown in Algorithm 11. We start by initialising the messages α_{ij}, ρ_{ij} and λ_{ik} to 0 and ω_{ij} to the median of S. Once the messages are initialised they are updated in turn with damping until convergence is achieved. Typically this is the case when the energy of the solution is stable over a few iterations. Outliers are determined as the ℓ points with the largest values of $\max_k(\lambda_{ik} + \omega_{ik})$. From the remaining points, the exemplars are then selected as the points for which $(\alpha_{ii} + \rho_{ii}) > 0$ is true. All other points i are assigned to the exemplar e satisfying $\text{argmax}_e(\alpha_{ie} + \rho_{ie})$.

5.3.3 Lagrangian duality (LR)

Another way to solve Eq. (5.1a) is via the Lagrangian dual function. The Lagrangian dual is generally a lower bound on the primal problem. However, in this

Algorithm 11: Affinity propagation with outlier selection

Input: Similarity matrix S and number of outliers ℓ
Output: Exemplars e, assignments a, and outliers o
`// Initialise messages`
foreach $i, j \in \{1, \ldots, N\}, k \in \{1, \ldots, \ell\}$ **do**
  $\alpha_{ij}, \rho_{ij}, \lambda_{ik} \leftarrow 0$
  $\omega_{ik} \leftarrow \text{median}(S)$
end
`// Perform clustering and outlier selection`
repeat
  $\rho_{ij} \leftarrow s_{ij} + \min\left[-\max_{t \neq j}(\alpha_{it} + s_{it}), -\max_t(\omega_{it})\right]$
  $\alpha_{ij} \leftarrow \begin{cases} \sum_{t \neq j} \max(0, \rho_{tj}) & i = j \\ \min\left[0, \rho_{jj} + \sum_{t \notin \{i,j\}} \max(0, \rho_{tj})\right] & i \neq j \end{cases}$
  $\lambda_{ik} \leftarrow \min\left[-\max_t(\alpha_{it} + s_{it}), -\max_{t \neq k}(\omega_{ti})\right]$
  $\omega_{ik} \leftarrow -\max_{t \neq i}(\lambda_{tk})$
until *convergence*
`// Extract final solution`
$o \leftarrow \ell$ points for which $\max_k(\lambda_{ik} + \omega_{ik})$ holds
$e \leftarrow$ points for which $(\alpha_{ii} + \rho_{ii}) > 0$ holds
$a_i \leftarrow$ assign point i to exemplar e satisfying $\text{argmax}_e(\alpha_{ie} + \rho_{ie})$
return e, a, o

case equality can also be proved. Additionally, optimising the dual is easier as the function is convex and thus has a unique optimum. The dual though is not differentiable and hence we rely on subgradient based methods.

Dualising the primal cost function Eq. (5.1a) by relaxing the constraint in Eq. (5.1c) with Lagrange multipliers λ, we obtain the following Lagrangian dual function:

$$\min \underbrace{\sum_i (1 - o_i)\lambda_i}_{\text{outliers}} + \underbrace{\sum_j c_j y_j + \sum_i \sum_j (d_{ij} - \lambda_i) x_{ij}}_{\text{clustering}} . \tag{5.11a}$$

subject to

$$x_{ij} \leq y_i \tag{5.11b}$$

$$\sum_k o_k = \ell \tag{5.11c}$$

$$0 \leq x_{ij}, y_j, o_k \leq 1 \quad \forall i, j, k \tag{5.11d}$$

We find valid assignments that attempt to minimise Eq. (5.11a) following the ideas of Bertsimas and Weismantel (2005). The Lagrange multipliers λ act as a penalty incurred for constraint violation which we try to minimise. From Eq. (5.11a) we see that the penalty influences two parts: outlier selection and

clustering. We pick good outliers by selecting the ℓ points with largest λ as outliers, as this removes a large part of the penalty. For the remaining $N - \ell$ points we determine clustering assignments by setting $x_{ij} = 0$ for all pairs for which $d_{ij} - \lambda_i \geq 0$. To select the exemplars we compute:

$$\mu_j = c_j + \sum_{i:d_{ij}-\lambda_i<0} (d_{ij} - \lambda_i), \tag{5.12}$$

which represents the amortised cost of selecting point j as exemplar and assigning points to it. Thus, if $\mu_j < 0$ we select point j as an exemplar and set $y_j = 1$, otherwise we set $y_j = 0$. Finally, we set $x_{ij} = y_j$ if $d_{ij} - \lambda_i < 0$. From this complete assignment we then compute a new subgradient $s^{[t]}$ and update the Lagrange multipliers $\lambda^{[t]}$ as follows:

$$s_j^{[t]} = 1 - \sum_j x_{ij} \tag{5.13}$$

$$\lambda_j^{[t]} = \max(\lambda_j^{[t-1]} + u^{[t]} s_j^{[t]}, 0), \tag{5.14}$$

where $u^{[t]}$ is the step size at time t computed as

$$u^{[t]} = u^{[0]}\alpha^t \quad \alpha \in (0, 1), \tag{5.15}$$

which is guaranteed to converge (Bertsimas and Weismantel, 2005) if a step function is used for which the following holds:

$$\sum_{t=1}^{\infty} u^{[t]} = \infty \qquad \text{and} \qquad \lim_{t\to\infty} u^{[t]} = 0, \tag{5.16}$$

which is the case for the step function in Eq. (5.15). To obtain the final solution we repeat the above steps until the changes become small enough, at which point we extract a feasible solution.

The pseudo code in Algorithm 12 shows how to compute the assignment matrix X, exemplars y, and outliers o with the method described above. The input required is the current Lagrange multipliers λ, which are first processed to find the outliers which correspond to the points with the largest λ values. Then a score indicating the quality of each point as an exemplar is computed. Based on this score we perform exemplar selections as well as cluster assignments.

Optimality of the Lagrangian duality-based method

In general the Lagrangian dual is a lower bound to the primal problem. However, as outlined in Section 2.3.1, under certain circumstances the solution to the La-

Algorithm 12: Lagrange relaxation iteration(λ)

Input: Lagrange multipliers λ
Output: Assignment matrix X, exemplar selections y, outliers o

$o \leftarrow \emptyset$ // Outlier indicators
$y \leftarrow \mathbf{0}$ // Exemplar indicators
$X \leftarrow \mathbf{0}$ // Assignments
// Selecting outliers
foreach $i \in \{1, \ldots, N\}$ **do**
  if λ_i *one of the* L *largest values* **then**
    $o \leftarrow o \cup i$
  end
end
// Compute exemplar scores
foreach $j \in \{1, \ldots, N\}$ **do**
  $\mu_j \leftarrow c_j$
  foreach $i \in \{1, \ldots, N\}$ **do**
    if $(dist(i,j) - \lambda_i) < 0$ **then**
      $\mu_j \leftarrow \mu_j + (dist(i,j) - \lambda_i)$
    end
  end
end
// Select exemplars and perform assignments
foreach $j \in \{1, \ldots, N\}$ **do**
  if $\mu_j < 0$ **then**
    $y_j \leftarrow 1$
  else
    $y_j \leftarrow 0$
  end
  foreach $i \in \{1, \ldots, N\}$ **do**
    if $(dist(i,j) - \lambda_i) < 0 \wedge y_j = 1$ **then**
      $x_{ij} \leftarrow 1$
    end
  end
end
return X, y, o

grangian dual can be equivalent to the primal solution. In the following we show that this is the case for the problem of clustering with outlier detection. Consider the constraint set $L = \{(x, y, o) \in \mathbb{R}^{n^2+2n} | x_{ij} \leq y_j \wedge \sum_i o_i \leq \ell \; \forall \; i, j\}$. Then the optimal value of the Lagrangian relaxation problem FLO_{LR} is equal to the cost of the following optimisation problem:

$$\text{minimise} \quad \sum_j y_j c_j + \sum_i \sum_j x_{ij} d_{ij} \tag{5.17a}$$

$$\text{subject to} \quad o_i + \sum_j x_{ij} = 1 \tag{5.17b}$$

$$x_{ij}, y_j, o_i \in conv(L) \tag{5.17c}$$

where $conv(L)$ is the convex hull of the set L. We now show that L is integral, i.e.:

$$conv(L) = \left\{(x, y, o) \in \mathbb{Z}^{n^2+2n} | x_{ij} \leq y_j \wedge \sum_i o_i \leq \ell \; \forall \; i, j\right\} \tag{5.18}$$

This in turn will imply that the solution of FLO_{LR} is equivalent to the linear program FLO_{LP}. In order to show that L is integral, we establish that the constraint matrix corresponding to the set L is totally unimodular (TU). For completeness, we recall several important definitions and theorems from integer program theory (Bertsimas and Weismantel, 2005; Schrijver, 1998):

Definition 1. *A matrix A is totally unimodular if the determinant of every square submatrix of A has value* $-1, 0$ *or* 1.

An equivalent definition of total unimodularity and often easier to establish is captured in the following definition.

Definition 2. *A matrix A is TU iff for any subset of rows X of A, there exists a colouring, or partition, of rows of X, with 1 or -1 such that the weighted sum of every column (while restricting the sum to rows in X) is -1, 0 or 1.*

Theorem 2. *Let A be a totally unimodular matrix and b an integral vector. Then the polyhedron* $P = \{x : Ax \leq b\}$ *is integral.*

Proof. Let $F = \{x : A'x = b'\}$ be a minimal face of P, where $A'x \leq b'$ is a subsystem of $Ax \leq b$ and A' has full row rank. Then we may permutate the coordinates in such a way that $A' = [U \quad V]$ for some unimodular matrix U, and $\begin{bmatrix} U^{-1}b \\ \mathbf{0} \end{bmatrix}$ is an integral vector in F. □

From this it follows that every linear program with integer data and totally unimodular matrix has an integral optimal solution.

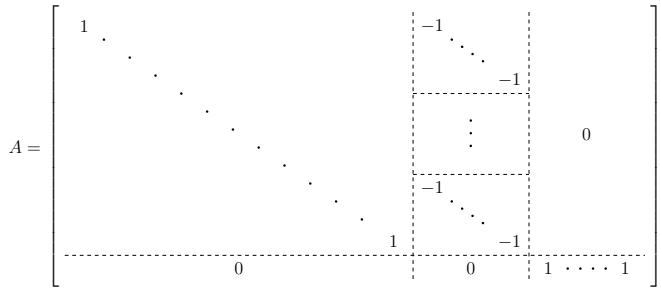


Figure 5.2: Visualisation of the building blocks of the A matrix. The top left is a $n^2 \times n^2$ identity matrix which is followed by n row stacked blocks of $n \times n$ negative identity matrices. To the right of those is another $n^2 \times n$ block of zeros. The final row in the block matrix consists of $n^2 + n$ zeros followed by n ones.

Theorem 3. *Let A be a totally unimodular matrix and let b, c be integral vectors. Then both problems in the LP-duality equality*

$$\max\{x^T x : Ax \leq b\} = \min\{y^T b : y \geq 0 \text{ and } yA = c\} \tag{5.19}$$

have integral optimal solutions.

Proof. Follows from Theorem 2, using the fact that the matrix

$$\begin{bmatrix} I \\ A^T \\ -A^T \end{bmatrix} \tag{5.20}$$

is also totally unimodular. □

With these definitions we are now ready to state and prove the main theorem:

Theorem 4. *Let $L = \{(x, y, o) \in \mathbb{Z}^{n^2+2n} | x_{ij} \leq y_j \wedge \sum_i o_i \leq \ell \ \forall \ i, j\}$ be the constraint set corresponding to the Lagrangian relaxation problem then L is totally unimodular.*

Proof. Consider the constraints

$$x_{ij} \leq y_j \ \forall \ i, j \tag{5.21a}$$

$$\sum_{i=1}^{n} o_i \leq \ell \tag{5.21b}$$

$$x_{ij}, y_j, o_i \in conv(L) \tag{5.21c}$$

We can express the above constraints in the form $Au = b$ where u is the vector:

$$u = [x_{11}, \ldots, x_{1n}, \ldots, x_{n1}, \ldots, x_{nn}, y_1, \ldots, y_n, o_1, \ldots, o_n]^T \tag{5.22}$$

and A is a block matrix of the form:

$$A = \begin{bmatrix} I & B & 0 \\ 0 & 0 & \mathbf{1} \end{bmatrix}. \tag{5.23}$$

Here I is an $n^2 \times n^2$ identity matrix, B is a stack of n matrices of size $n \times n$ where each element of the stack is a negative identity matrix, and $\mathbf{1}$ is a $1 \times n$ block of 1's. See Figure 5.2 for a detailed visualisation.

Using Definition 2 we now prove that A is TU. Take any subset Z of rows of A. Whether we colour the rows of Z with 1 or -1, the column sum (within Z) of a column of I will be $1, 0$ or -1. A similar argument holds for columns of the block matrix $\mathbf{1}$. Now consider the submatrix B. We can express X as

$$Z = \cup_{i=1, i \in B(Z,:)}^{n} Z_i, \tag{5.24}$$

where each $Z_i = \{r \in Z | Z(r, i) = -1\}$. Given that B is a stack of negative diagonal matrices, $Z_i \cap Z_j = \emptyset$ for $i \neq j$. Now consider a column j of B. If Z_j has even number of $-1'$s, then split the elements of Z_j evenly and colour one half with 1 and the other with -1. Then the sum of column j (for rows in Z) will be 0. On the other hand, if another set of rows Z_k has odd number of -1, colour the rows of Z_k alternating with 1 and -1. Since Z_j and Z_k are disjoint their colouring can be carried out independently. Then the sum of column j will be 1 or -1. Thus we satisfy the condition of Theorem 3 and conclude that A is TU. □

Scalable implementation details

In order to enable the algorithm to scale to large datasets we need to consider the limited availability of computational resources. Most importantly, we cannot assume that the complete distance matrix can fit into the main memory. Therefore, we compute the distances between point pairs on the fly. Since this involves N^2 evaluations per iteration it is the most costly part of the method. However, the evaluation of the distance function can be easily parallelised as they are independent of each other. In practice with simple distance functions, such as the Euclidean distance, approximately 75% of the computational time is spent evaluating the distance function. This percentage will become larger with more complex distance functions. Another important point is that just as storing the full distance matrix is not possible, neither is storing the full assignment matrix X. However, we are only interested in the values where $x_{ij} = 1$, which is a small portion of the full matrix. Thus we can use standard sparse matrix implementations to manage the assignment matrix.

5.3.4 Comparisons

Now that we have presented the different methods we want to give an overview of the advantages and disadvantages of these. Table 5.1 presents a quick overview of different properties of the proposed methods which we will discuss in more detail. First, the speed of APOC and LR clearly outperforms LP by a large margin, however, other methods such as k-means-- still are faster.

With regards to optimality we know that the optimal solution of the Lagrangian relaxation FLO_{LR} is bounded by the optimal solutions of the linear program FLO_{LP} and integer program FLO_{IP}, i.e.:

$$FLO_{LP} \leq FLO_{LR} \leq FLO_{IP} \tag{5.25}$$

(Bertsimas and Weismantel, 2005). Thus, if LP finds a solution that is integer it is equivalent to the IP one, i.e. $FLO_{LP} = FLO_{IP}$. The LR implementation sacrifices optimality for performance by using discounted updates of the solution matrix X as well as stopping before converging to a pure integer solution. APOC has no theoretical guarantees on either convergence or optimality bounds of the solution. While such guarantees can be given for certain types of structures with belief propagation (Weiss et al., 2007) it is unclear how they apply to the special case of APOC. However, in practice both APOC and LR achieve scores very close to the optimum.

The speed of LP clearly makes it impossible to scale for large datasets with several thousands of data points, which leaves us with the comparison of APOC and LR. APOC requires storage for messages and similarities and performs operations which cannot be supported by standard sparse matrix implementations. As such APOC cannot be made truly scalable. LR on the other hand spends the majority of its runtime computing distances between pairs of points, a task that can easily be parallelised. Furthermore the actual assignment matrix can be stored using standard sparse matrix implementations. Thus LR is a much better candidate for large scale datasets. Finally, with regards to extensibility LP is the easiest as the relaxation only modifies the value range of variables. LR is more involved as there is more freedom in the relaxation of the constraints and the dual problem needs to be efficiently solved. Affinity propagation is the hardest modify as it requires very careful choice of constraints which can be modelled by the graphical model and laborious derivation of update rules, which makes it difficult to derive solutions to new problems.

	LP	APOC	LR
Speed	$--$	$+$	$+$
Optimality	$++$	$+$	$+$
Scalability	$--$	$-$	$+$
Extensibility	$++$	$-$	$+$

Table 5.1: Advantages and disadvantages of LP, APOC and LR. See Section 5.3.4 for detailed explanation.

5.4 Experiments

In this section we evaluate the proposed methods on both synthetic and real data. First, we present experiments using synthetic data to show different properties of the methods and perform quantitative analysis. Next, we process hurricane GPS traces to show the applicability of this method to temporal data. In the last two experiments we process MNIST images as well as images from the "New College" dataset (Smith et al., 2009).

Both APOC and LR require a cost for creating clusters. In all experiments we obtain this value as $\theta * \text{median}(x_{ij}), \theta \in [1, 30]$, i.e. the median of all distances multiplied by a scaling factor θ which gives equal importance to all points during exemplar selection. Varying this value impacts the result in a predictable manner, i.e. a smaller value results in more clusters, though any value in the provided range will obtain a sensible result.

5.4.1 Synthetic data

We use synthetic datasets to evaluate the performance of the proposed methods in a controlled setting. We randomly sample k clusters with m points each from d-dimensional normal distributions $\mathcal{N}(\mu, \Sigma)$ with randomly selected μ and Σ. To these clusters we add ℓ additional outlier points that have a low probability of belonging to any of the selected clusters.

We compare APOC and LR against k-means-- using k-means++ (Arthur and Vassilvitskii, 2007) to select the initial centres. Euclidean distance is used as the metric for all methods. Unless mentioned otherwise k-means-- is provided with the exact number of clusters generated, while APOC and LR are required to determine this automatically.

To assess the performance of the methods we use the following three metrics:

1. Normalised Jaccard index, see Section 2.6.3, measures how accurately a method selects the ground truth outliers. It is a coefficient computed between selected outliers O and ground-truth outliers O^*. The final coefficient

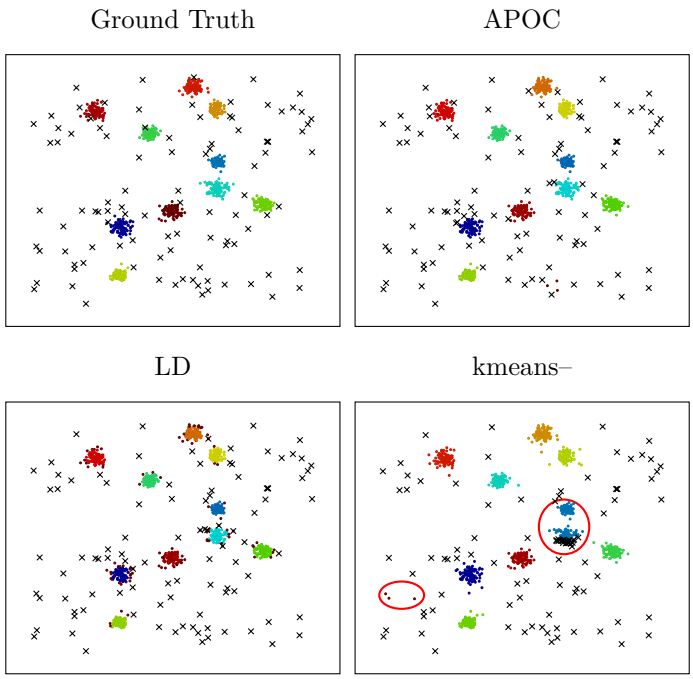


Figure 5.3: Clustering results for a 2D dataset with $k = 10$, 100 points per cluster and 100 outliers. APOC and LR accurately detect the clusters and select appropriate outliers without requiring k as input. k-means--, provided with the correct value of k, fails to split certain clusters which results in the selection of non-outliers as outliers. Black crosses indicate the outliers selected, points of identical colour indicate clusters and the red circles indicate errors made by k-means--.

is normalised with regards to the best possible coefficient obtainable in the following manner:

$$J(O, O^*) = \frac{|O \cap O^*|}{|O \cup O^*|} \Big/ \frac{\min(|O|, |O^*|)}{\max(|O|, |O^*|)}. \tag{5.26}$$

2. Local outlier factor (Breunig et al., 2000) (LOF), see Section 2.6.2, measures the outlier quality of a point. We compute the ratio between the average LOF of O and O^*, which indicates the quality of the set of selected outliers.

3. V-Measure (Rosenberg and Hirschberg, 2007), see Section 2.6.1, indicates the quality of the overall clustering solution. The outliers are considered as an additional class for this measure.

For all the metrics a value of 1 is the optimal outcome.

In Figure 5.3 we present clustering results obtained by the different methods. Both APOC and LR manage to identify the correct number of clusters and select accurate outliers. k-means-- on the other hand, even with good initialisation and correct value for k specified, fails to find the correct clusters, and as a result, finds

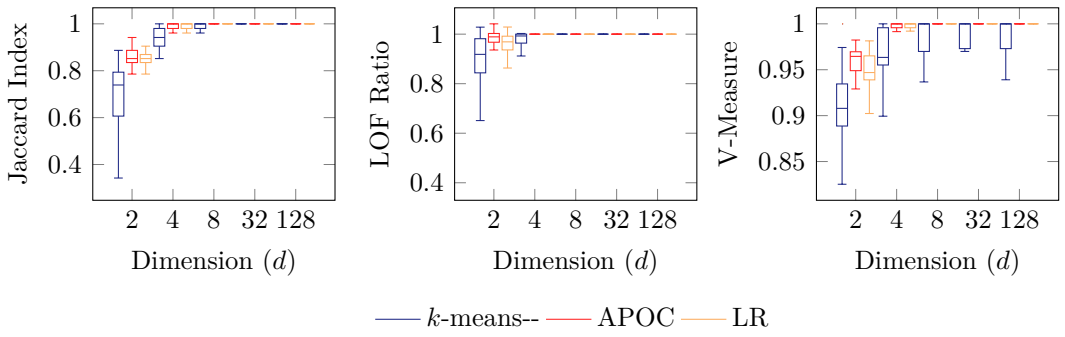


Figure 5.4: The impact of increasing the data dimensionality on the quality of the clustering and outlier selection quality. APOC and LR provide similar results while k-means--, provided with the correct k, has more trouble.

a suboptimal solution. The errors made by k-means-- are highlighted by the red circles.

We first investigate the influence of the data dimensionality on the results. In Figure 5.4 it is clear that in general the quality of the solution increases with higher dimensions. This can be explained by the fact that in higher dimensional spaces points are farther apart and hence easier to cluster. Looking at k-means-- we can see that it struggles more then the other two methods even though it is provided with the correct number of clusters. In higher dimensions it achieves perfect scores for the two outlier centric measures but is unable to always find the correct solution to the entire clustering problem. Looking at APOC and LR we observe that both have little trouble finding perfect solutions in high dimensions. In lower dimensions LR shows a bit more variability compared to APOC. In general, two dimensional data is the most challenging and thus will be used for all further experiments.

The number of outliers ℓ is a parameter that all methods require. Typically, the correct value of ℓ is unknown and it is therefore important to investigate how the algorithms react when the user's estimate is incorrect. We generate 2D datasets with 2000 inliers and $\ell^* = 200$ outliers and vary the number of outliers ℓ selected by the methods. The results in Figure 5.5 show that in general no method fails completely if the correct value for ℓ is not provided. Looking at the Jaccard index we see that if $\ell < \ell^*$ all methods select only outliers. When $\ell > \ell^*$ we can see a difference in performance, namely that LR selects the largest outliers which APOC does to some extent as well, while k-means-- does not seem to be very specific about which points to select. This difference in behaviour stems from the fact that APOC and LR optimise a cost function in which removing the biggest outliers is the most beneficial. Looking at the LOF results we can see that if we select a small value for ℓ the scores are low, which is a result of

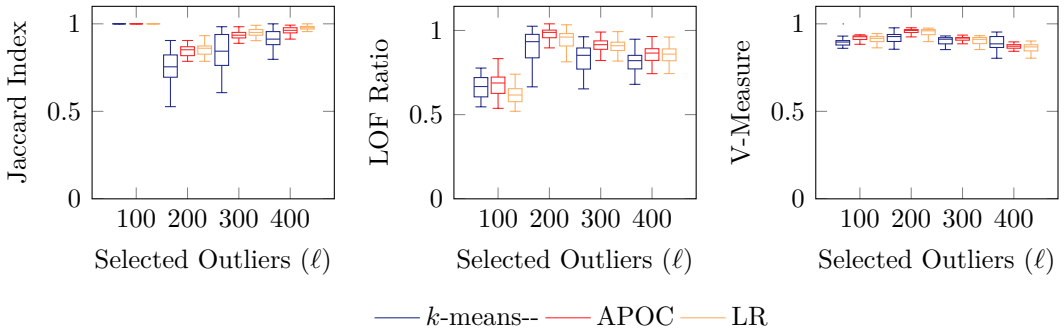


Figure 5.5: The impact of number of outliers specified (ℓ) on the quality of the clustering and outlier detection performance. APOC and LR perform similarly with more stability and better outlier choices compared to k-means--. We can see that overestimating ℓ is more detrimental to the overall performance, as indicated by the LOF Ratio and V-Measure, then underestimating it.

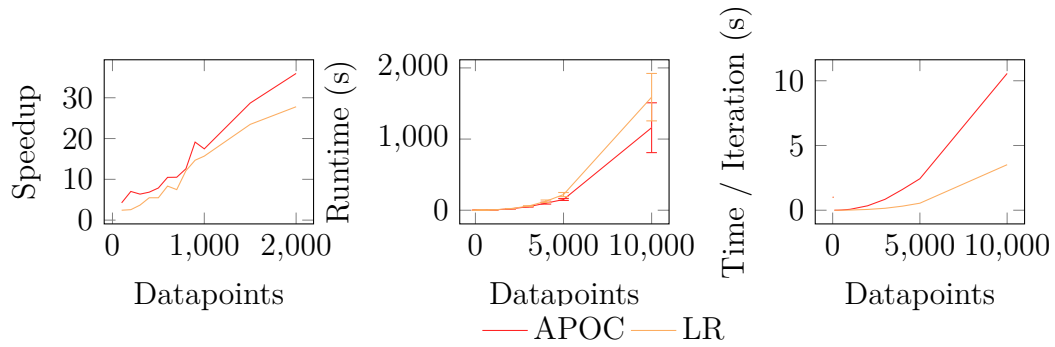


Figure 5.6: The graphs shows how the number of points influences different measures. In (a) we compare the speedup of both APOC and LR over LP. (b) compares the total runtime needed to solve the clustering problem for APOC and LR. Finally, (c) plots the time required (on a log scale) for a single iteration for APOC and LR.

the metric as opposed to a bad selection of outliers. Selecting more outliers then the dataset contains causes the quality to drop the higher the value of ℓ. This is to be expected as we select more and more points that are not true outliers, thus lowering the LOF score. The gradual decline of the score also indicates that we initially select true outliers and then progressively select points at the edge of clusters. LR and APOC again perform similarly with less variability and better scores compared to k-means--. Finally, V-Measure shows that the overall clustering results remain accurate even if the number of outliers is misspecified. For large differences between actual and specified outliers a drop in clustering performance can be observed. Again, as in the two other metrics, k-means-- exhibits larger variations when compared to APOC and LR which optimise the same underlying cost function.

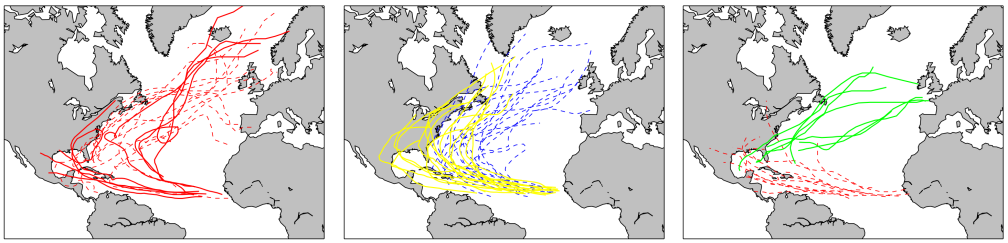
Since both APOC and LR are not guaranteed to find the optimal solution we

examine how close the solutions obtained are to the optimum. The ground truth needed for this is obtained by solving the LP formulation in Section 5.3.1 with CPLEX. This comparison indicates what quality can be typically expected from the two methods. Additionally, we can evaluate the speed of the two approximations. We evaluate 100 datasets, consisting of 2D Gaussian clusters and outliers, with varying number of points. On average APOC obtains an energy that is $96\% \pm 4\%$ of the optimal solution found by LP, LR obtains $94\% \pm 5\%$ of the LP energy while k-means--, with correct k, only obtains $86\% \pm 12\%$ of the optimum. These results reinforce the previous analysis; APOC and LR perform similarly while outperforming k-means--. We now look at the speedup of APOC and LR over LP, as shown in Figure 5.6 (left). Both methods outperform LP by a large margin which increases as more points are involved. Overall for a small price in accuracy the two methods obtain a solution significantly faster.

Next we compare the performance between APOC and LR. Figure 5.6 (middle) shows the overall runtime of both methods for varying number of data points. Here we observe that APOC takes less time then LR. However, by observing the time a single iteration takes, shown in Figure 5.6 c), we see that LR is much faster on a per iteration basis compared to APOC. In practice LR requires several times the number of iterations of APOC, which is affected by the step size function used. Using a more sophisticated method of computing the step size will provide large gains to LR. Finally, the biggest difference between APOC and LR is that the former requires all messages and distances to be held in memory. This obviously scales poorly to large datasets. Conversely, LR computes the distances during running time and only needs to store indicator vectors and a sparse assignment matrix, thus using much less memory. This makes LR amenable to processing large datasets. For example, with single precision floating point numbers, dense matrices and 10 000 points APOC requires around 2200 MB of memory while LR only needs 370 MB. Further gains can be obtained by using sparse matrices which is straight forward in the case of LR but complicated for APOC.

5.4.2 Hurricane dataset

In this experiment we use hurricane GPS trace data from 1970 to 2010 provided by the National Oceanic and Atmospheric Administration (NOAA). The data provides a time series of longitude and latitude coordinates for each storm forming a GPS trace. In order to compare the overall shape of these traces we use the discrete Fréchet distance (Eiter and Mannila, 1994). Intuitively this measures the minimal distance required to connect points on two curves being compared, i.e. structurally similar curves have a low score. Before computing the Fréchet distance between two curves we move their starting points to the same location.



(a) Outliers (b) Clusters (c) Clusters

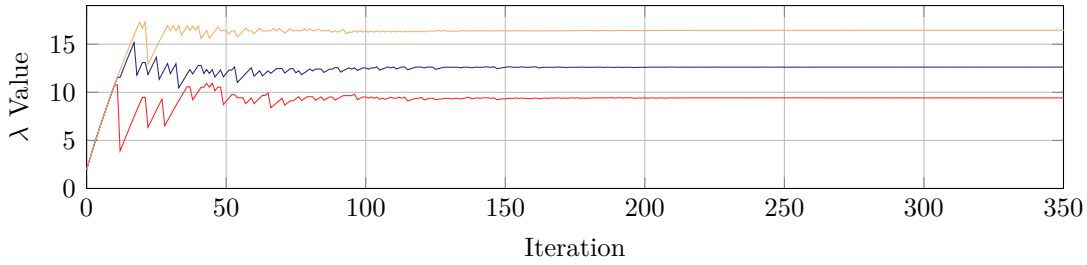
Figure 5.7: Outliers and clusters found in the hurricane data set. In (a) we show the outliers as two groups: Cape Verde-type hurricanes (thick lines) and others (dashed lines). In (b) and (c) we display examples of clusters which all exhibit similar shape albeit not necessarily in the same position.

This means that we are comparing their shapes and not their global location. Clustering the 700 traces using the LR method with $\ell = 20$ we obtain clusters that move in a similar direction, as shown in Figure 5.7 b) and c). Analysing the outliers shown in Figure 5.7 a) we find that half of them are category 4 and 5 Cape Verde-type hurricanes, shown by the thick stroke. The other half of the outliers were selected due to either their long life time, unusual motion, or high destructive power and are shown with dashed line. This demonstrates that the outliers found by the proposed methods reveal interesting patterns in spatial temporal data which are not directly apparent.

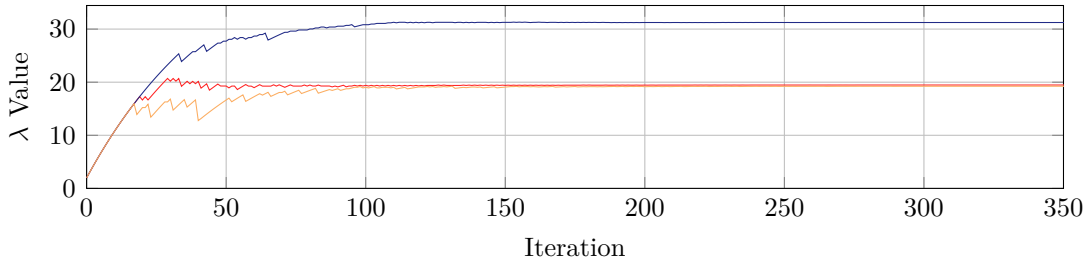
To better understand the behaviour of LR we plot the value of the λ values associated with outliers and exemplars in Figure 5.8. One can see how after about 100 iterations the values level out and remain stable. Interestingly the outliers have higher λ values compared to the exemplars which allows the method to distinguish between them more easily. Lastly, the outliers have smoother curves, i.e. less jumps, when compared to the exemplars. This indicates that outliers do not change significantly whereas exemplars will vary more during the optimisation process. Overall, stable results are achieved after about a third of the runtime suggesting that a more sophisticated termination criterion can be used to improve the rate of convergence.

5.4.3 MNIST dataset

The MNIST dataset, introduced by LeCun et al. (1998), contains 28×28 pixel images of handwritten digits. We extract features from these images by representing them as 768 dimensional vectors which we reduce to 25 dimensions using PCA. The point wise distances required by our methods is computed by the L^2 norm. In Figure 5.9 we show exemplary results obtained when processing 10 000 digits with the LR method with $\theta = 5$ and $\ell = 500$. Each row in Figure 5.9 a)



(a) Exemplars



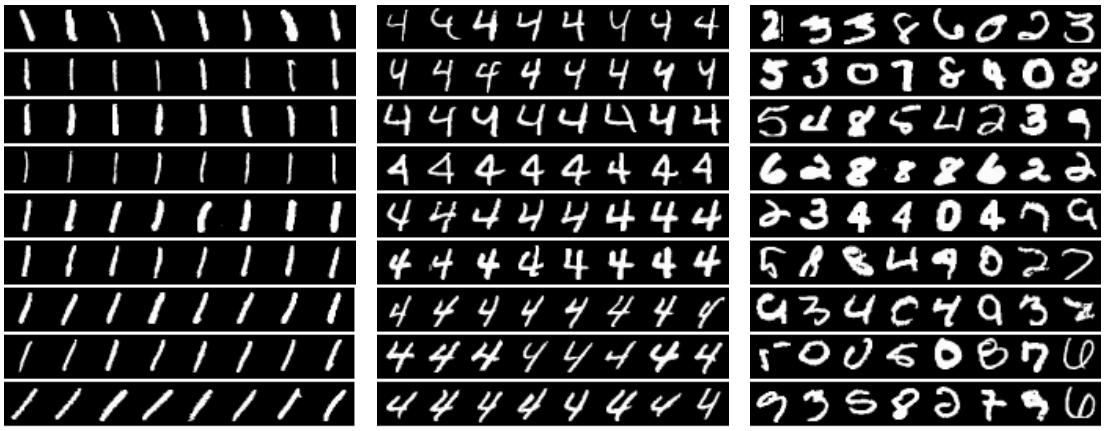
(b) Outliers

Figure 5.8: Evolution of the λ values of select exemplars and outliers over the iterations. In both cases we can observe how after about 150 iterations the values have stabilised. Additionally, outliers have larger λ values and exhibit less variation compared to exemplars.

and b) shows examples of clusters representing the digits 1 and 4 respectively. This illustrates how different the same digit can appear and the separation induced by the clusters. Figure 5.9 c) contains a subset of the outliers selected by the method. These outliers have different characteristics that make them sensible outliers, such as: thick stroke, incomplete, unrecognisable or ambiguous meaning.

To investigate the influence the cluster creation cost has we run the experiment with different values of θ. Table 5.2 shows results for values of cost scaling factor $\theta \in \{5, 15, 25, 50\}$ for both LR and APOC and $k \in \{10, 30\}$ for k-means. For LR there is a clear peak with $\theta = 15$ which produces the best results of any method. If the cluster creation cost is further increased, performance drops. APOC behaves differently as it seems to plateau over a wide range of values and only degrade performance with extreme values. That being said it never reaches the quality of the LR solution. With k-means we can see that the number of clusters makes a difference but that number may not reflect the true number of clusters expected.

Observing the results for LR and APOC the number of clusters is always decreasing for APOC, whereas it increases and decreases with increasing θ value for LR. This can indicate that LR is more capable of changing its solution drastically compared to APOC.



(a) Digit 1 (b) Digit 4 (c) Outliers

Figure 5.9: Each row in (a) and (b) shows a different appearance of a digit captured by a cluster. The outliers shown in (c) tend to have heavier then usual stroke, are incomplete or are not recognisable as a digit.

	LR				APOC				k-means--	
θ	5	15	25	50	5	15	25	50	n.a.	n.a.
V-Measure	0.52	0.67	0.54	0.44	0.53	0.54	0.54	0.48	0.52	0.58
Homogeneity	0.78	0.74	0.65	0.47	0.82	0.73	0.67	0.54	0.51	0.71
Completeness	0.39	0.61	0.46	0.42	0.39	0.43	0.46	0.45	0.52	0.49
Clusters	120	13	27	14	129	51	32	17	10	30

Table 5.2: Evaluation of clustering results of the MNIST data set with different cost scaling values θ for LR and APOC as well as different settings for k-means--. We can see that increasing the cost results in fewer clusters but as a trade off reduces the homogeneity of the clusters.

5.4.4 Outliers in image dataset

In this section we show the results obtained from performing clustering with outlier selection on typical robotics data. We use data from the "New College" dataset (Smith et al., 2009) as well as data collected on the University of Freiburg campus. Our goal in this experiment is to group images based on the type of scenery they represent. As such we are not splitting images into smaller patches but use the entire image directly. Each image is described by an HSV colour and LBP texture histogram with the distance between histograms computed using the Bhattacharyya distance, i.e.:

$$dist(\mathcal{I}_a, \mathcal{I}_b) = d_B(\mathcal{H}_{HSV}(a), \mathcal{H}_{HSV}(b)) + d_B(\mathcal{H}_{LBP}(a), \mathcal{H}_{LBP}(b)), \tag{5.27}$$

where $dist(\mathcal{I}_a, \mathcal{I}_b)$ is the distance between two images, $d_B(\cdot, \cdot)$ the Bhattacharyya distance between two histograms, $\mathcal{H}_{HSV}$ the HSV colour histogram extractor and

LR	APOC	k-means--
0.80 ± 0.08	0.84 ± 0.14	0.37 ± 0.08

Table 5.3: Average and standard deviation of the percentage of true outliers correctly identified by the different methods. LR and APOC clearly outperform k-means--. APOC has slightly higher accuracy while LR has lower variability in the results.

$\mathcal{H}_{LBP}$ the local binary pattern histogram extractor.

A big challenge with this type of data is obtaining accurate ground truth. For this reason we sample a number N of images belonging to the "New College" dataset and inject a small number ℓ of images from the "Freiburg" dataset. These injected images will act as the outliers since they clearly do not belong to the "New College" dataset. We also do the inverse, i.e. using "Freiburg" as the main dataset and sample outliers from the "New College" data set. This allows us to obtain accurate ground truth while still using images typically encountered in robotics, even if the data is from two different datasets. We sample multiple data sets processed with k-means--, APOC and LR and measure the accuracy of the methods at identifying the correct outliers. The average and standard deviation of the three methods on data sets with $N = 2000$ and $\ell = 100$ is shown in Table 5.3. All methods were provided with the correct value of ℓ and k-means additionally was provided with $k = 15$, which is a typical number of clusters found by APOC and LR in these data sets. From the results we can clearly see how k-means-- has trouble identifying good solutions which is mainly due to the way clusters and outliers are more rigidly defined compared to APOC and LR. With APOC and LR we can see that they perform on similarly. APOC has slightly higher accuracy but has higher variability compared to LR. From this we can conclude that even in complex image data with simple features the proposed methods are capable of selecting the majority of the true outliers added. The fact that the algorithm is capable of identifying visually distinct outliers even if they form clusters of their own demonstrates the quality of the methods. The output generated for a run with inliers drawn from the "New College" dataset and outliers from the "Freiburg" dataset is shown in Figure 5.11. The images in the bottom right show samples of the outliers while the other groups show members of the different clusters. The outliers clearly belong to the "Freiburg" dataset. Investigating the clusters themselves we can see how the clusters represent different thematic groups encountered in the "New College" dataset.

Another way to demonstrate the quality of the outliers is to remove images of a certain type from the original dataset and then slowly add them back to the dataset and observe how the outliers change. The result of doing this is shown

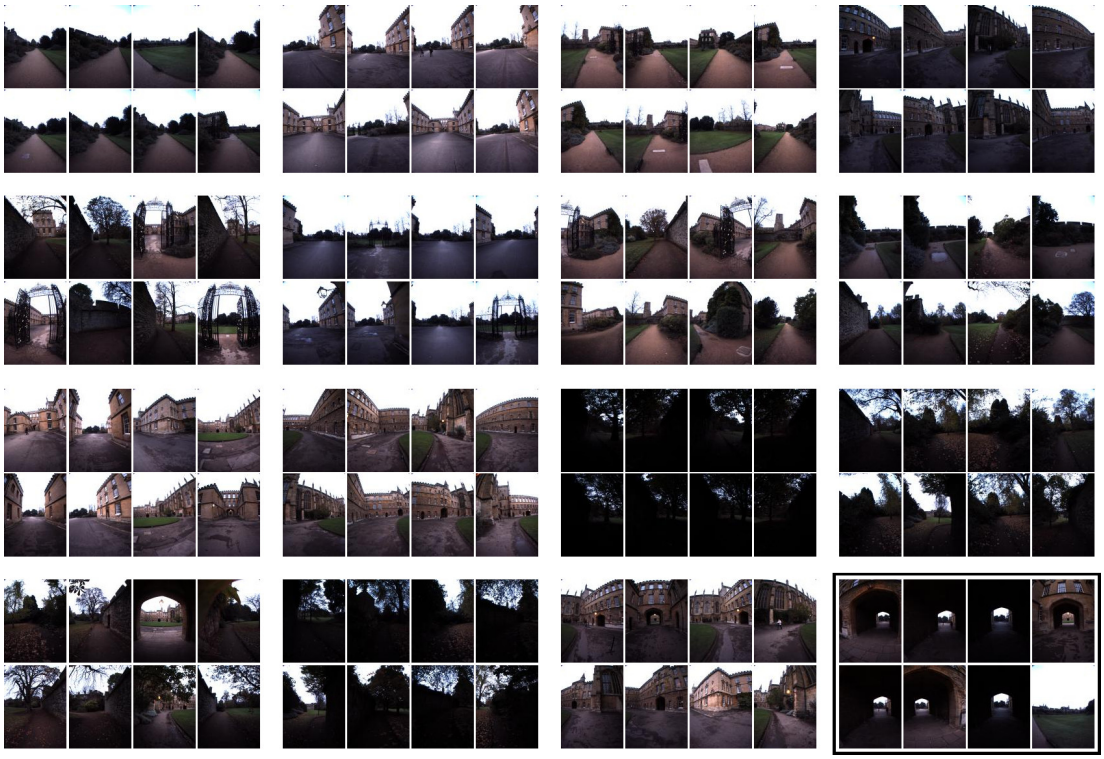


Figure 5.10: Samples of clusters and outliers (bottom right) when images from the "New College" dataset are processed. The clusters capture distinct aspects while the outliers, consisting mainly of arch ways show an uncommon and highly distinct concept in the dataset.

in Figure 5.12 where we remove all images containing lawns and gravel walk ways and then over time add them back. The left most image shows the outliers selected when all images of the above type have been removed. The middle image shows the outliers once a handful of the removed images are added back in. As we can see they are clearly represented as outliers. Finally, the right most image shows the outliers when all removed images have been added back. The clear outliers in the middle image have vanished as they have formed their own cluster and the outliers are mostly back to the type seen in the left most image.

Finally, we show the results obtained for clustering images belonging to the "New College" dataset. The results are shown in Figure 5.10. The outliers are shown in the bottom right surrounded by a border while the remaining groups show members of the different clusters. We can see that most of the outliers consist of images close to or inside an arch way. These images have very peaked histograms compared to other images in the dataset. Looking at the clusters we can see thematically different groups, some of which a human would possibly merge into a single cluster. The important aspect here is that the outliers tell us that either more similar images are needed or they can be ignored as they could otherwise cause issues.

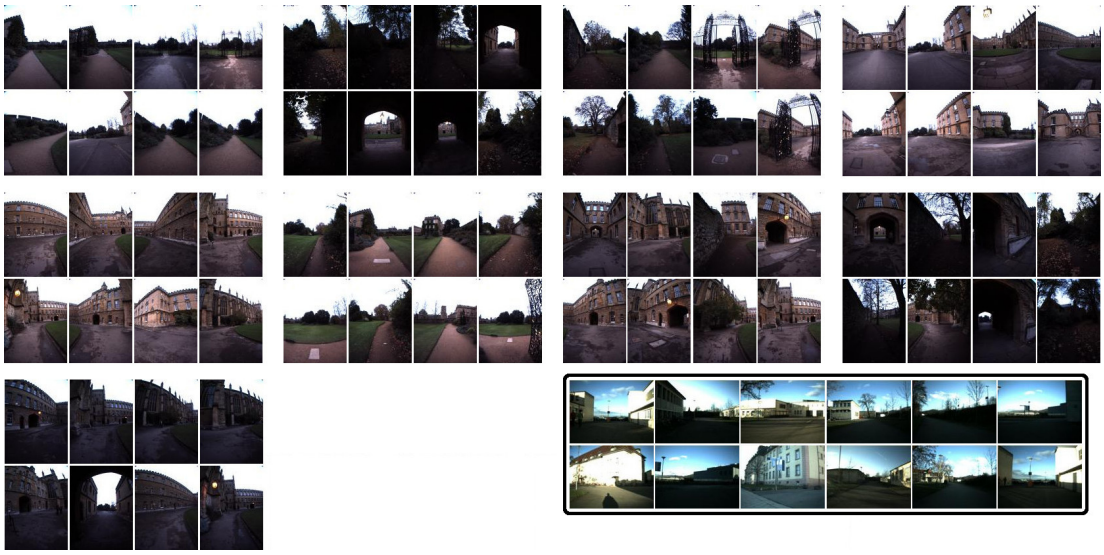


Figure 5.11: Outliers and samples of cluster members selected by APOC. All images belonging to the "Freiburg" dataset have been selected as outliers.

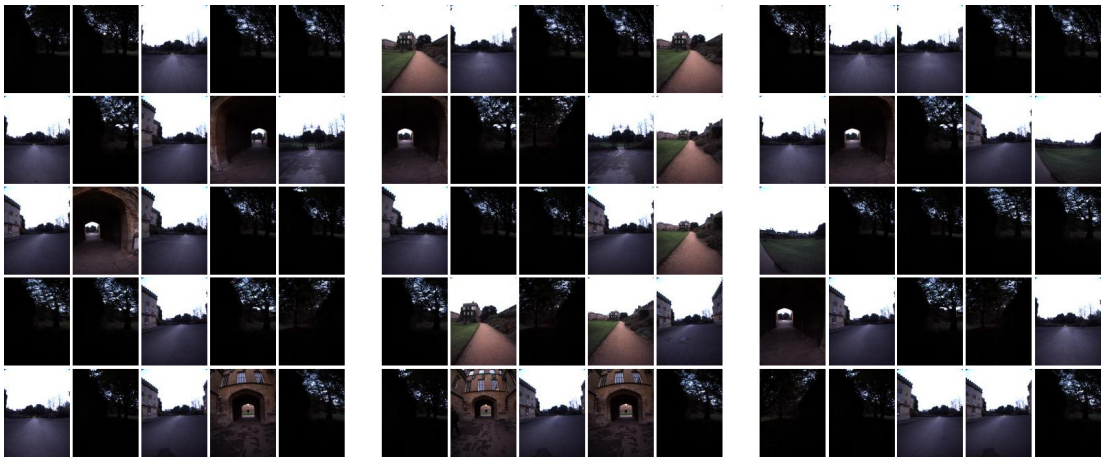


Figure 5.12: Effect of adding a novel type of observation into the system over time. The left image are the outliers before new observations are added, the middle image shows the outliers when a few new observations have been added and the right image are the outliers once all observations have been added.

5.5 Summary

In this chapter we presented a novel approach to joint clustering and outlier selection formulated as an integer program. The proposed optimisation problem automatically selects the number of clusters while jointly selecting a fixed number of outliers. We described two algorithms to solve the optimisation problem: (i) an extension of affinity propagation to outlier selection and (ii) a Lagrangian duality based relaxation method. For the Lagrangian relaxation based method we provide a proof of the optimality of the solution, something that is missing in affinity propagation. In the experiments on synthetic data we show how the joint optimisation outperforms two stage approaches such as k-means--. The results also highlight the overall quality of the solutions and resistance to misspecification of the number of outliers. Additionally, comparing the solutions of APOC and LR to the optimal solution obtained via linear programming we observe that the solutions are close to the optimum at a fraction of the computational time. In the next experiments the methods are applied to real data demonstrating their ability to cluster and select outliers. In the hurricane dataset the outliers capture a class of devastating hurricanes with no apparent pattern. Evaluation on the popular MNIST dataset indicates the capacity to find visually distinct outliers. Finally, the methods are applied to image data from the "New College" dataset which demonstrates the power of the method to identify atypical images in a dataset.

Chapter 6

Conclusion

This thesis addressed the problem of how to enable a robot to build and maintain a model of the environment in a fully unsupervised manner. This problem was approached using affinity propagation, a clustering algorithm, which was extended to solve challenges faced by such methods in robotics applications. This results in methods capable of learning an environment model in real-time while the robot moves around the environment, clustering data from multiple sensors only requiring information on a per sensor basis, and clustering data while jointly selecting the most likely outliers. While the methods are used in the context of model learning for long-term autonomy they are general in nature and, as such, can be applied to a variety of clustering tasks. All methods are based on combinatorial optimisation techniques allowing the methods to automatically select the number of clusters. The optimisation problems are then solved using either belief propagation in the case of affinity propagation, or Lagrangian duality in the case of FLO. This chapter concludes the thesis with a summary of the contributions as well as some potential directions for future work.

6.1 Summary of contributions

The contributions in this thesis relate to real-time self-supervised model learning, clustering of multiple data sources based solely on per sensor information and clustering with outlier selection. The following gives a short summary of each of these contributions.

6.1.1 Self-supervised learning

In Chapter 3 we developed a system that allows a robot to build and maintain a model of the environment in a self-supervised manner in real-time with no prior knowledge. The system uses affinity propagation to cluster observations which creates the foundation of the model. By adding self-supervision, in the form of collision information obtained from bumpers, the robot can additionally learn which clusters represent obstacles and which do not. This allows the system to build and maintain models for safe navigation in completely unknown envi-

ronments. In order to obtain real-time performance, the scalability of affinity propagation was addressed with an extension, meta-point affinity propagation, capable of clustering large amounts of data in real-time. The applicability of this system to long-term autonomy was shown in experiments demonstrating the method successfully reutilised parts of a model learnt from data collected months ago to model changes in the environment. Additionally, the flexibility of the system was demonstrated by applying it to the task of learning a model capable of predicting collisions from laser scans with only bump sensor information available for self-supervised training.

6.1.2 Multi-sensor clustering

A typical issue in robotics is that of combining data from multiple sensors into one cohesive model. This often involves tuning parameters which weight the importance of the different sensors in the final model which is time intensive and difficult to generalise. In Chapter 4 we presented a solution to this problem which required the user to only provide information about each sensor individually while the method selected the best combination of the data with regards to the clustering solution. Experiments on RGB-D data demonstrated the effectiveness of combining both data modalities with automated fusion of data sources.

6.1.3 Clustering with outliers

Chapter 5 addressed the important problem of dealing with noisy observations in the data. Instead of purely removing noise, the method selected the ℓ most appropriate outliers and returned them together with the clusters. The process of joint clustering and outlier selection is formulated as an integer program to which two solutions were presented. The first one is a generalisation of affinity propagation to clustering with outlier selection while the second one is based on the Lagrangian duality formulation of the integer program. The Lagrangian duality algorithm is provably optimal and can be easily parallelised. Extensive experiments on synthetic and real datasets demonstrated the capabilities of both methods at revealing interesting outliers and obtaining high quality clusters.

6.2 Future work

In this section we provide ideas for future research directions based on the methods and results presented in this thesis.

6.2.1 Long-term autonomy

While the method for self supervised model learning presented in Chapter 3 can automatically learn a model of the environment it currently does not integrate planning and exploration aspects, which would make it a truly autonomous system. Adding exploration and planning into the system would allow the robot to improve the model autonomously. Furthermore, integrating the outlier information provided by methods presented in Chapter 5 would be a good starting point to obtain information about the type of observations the robot needs to gather.

6.2.2 Multi-sensor clustering

Layered affinity propagation, introduced in Chapter 4, currently requires all data sources to observe the same area at the same time. Exploring ways to remove this constraint would increase the flexibility of the method significantly. Another possibility to exploit multiple data sources is to learn models which predict expected observations of another sensor. As predicting raw sensor readings is impractical predicting the type of observation is more interesting. One possible application for such predictions is to extrapolate from high density data close to the robot to sparse data far away from the robot in applications such as terrain or object classification.

6.2.3 Lagrangian duality

Lagrangian duality offers many possibilities for future developments. On an implementation side, improvements to the scalability of the FLO algorithm introduced in Chapter 5 by exploiting multithreading and GPU computing. Currently the method requires the user to provide the number of outliers ℓ to be selected. For some applications this can be seen as a restriction, however, reformulating the original integer program should allow the removal of this requirement.

In general the properties of Lagrangian duality make it very appealing for future research as encoding constraints is straightforward and the resulting optimisation problem is always convex. In this vein, investigating subgradient optimisation with regards to the impact different step size functions and subgradient update schemes have on convergence speed and quality of the final solution an interesting are for further investigation. Additionally, exploring ways to incrementally add data to the optimisation in a theoretically sound manner can be beneficial in many online applications.

6.2.4 Clustering methods

With regards to clustering techniques, incorporating data density information directly into the clustering process can be beneficial as demonstrated by DBSCAN (Ester et al., 1996). In the methods developed in this thesis the density of data points is not directly considered during the clustering. One simple solution is to apply a transformation to the distances which yields new distance values that take density into account. However, having a method capable of automatically considering distance as well as density is a preferable solution.

Finally, relaxing the restrictions of fully unsupervised operation opens up an entirely new area where small amounts of data could be used to bootstrap a system. One example would be the conditioning of the clustering method with labelled data to ensure the creation of specific types of clusters. Such methods would allow having the flexibility of an unsupervised method with the control of a supervised method.

Bibliography

R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Susstrunk. SLIC Superpixels Compared to State-of-the-art Superpixel Methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11):2274–2282, 2012.

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

D. Arthur and S. Vassilvitskii. k-means++: The Advantages of Careful Seeding. In *Proc. of the ACM-SIAM Symposium on Discrete Algorithms*, 2007.

S. Bay and M. Schwabacher. Mining distance-based outliers in near linear time with randomization and a simple pruning rule. In *Proc. of the International Conference on Knowledge Discovery and Data Mining*, 2003.

R. Bekkerman and J. Jeon. Multi-modal Clustering for Multimedia Collections. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007.

D. Bertsekas. *Nonlinear Programming*. Athena Scientific, 1999.

D. Bertsimas and R. Weismantel. *Optimization over Integers*. Dynamic Ideas Belmont, 2005.

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

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2009.

M. Breunig, H. Kriegel, R. Ng, and J. Sander. LOF: Identifying Density-Based Local Outliers. In *Int. Conf. on Management of Data*, 2000.

F. Cao, M. Ester, W. Qian, and A. Zhou. Density-Based Clustering over an Evolving Data Stream with Noise. In *SIAM International Conference on Data Mining*, 2006.

V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM Computing Surveys*, 41(3):15, 2009.

M. Charikar, S. Khuller, D.M. Mount, and G. Narasimhan. Algorithms for Facility Location Problems with Outliers. In *Proc. of the ACM-SIAM Symposium on Discrete Algorithms*, 2001.

S. Chawla and A. Gionis. k-means--: A Unified Approach to Clustering and Outlier Detection. In *SIAM International Conference on Data Mining*, 2013.

S. Chawla and P. Sun. SLOM: A new measure for local spatial outliers. *Knowledge and Information Systems*, 9(4):412–429, 2006.

K. Chen. A constant factor approximation algorithm for k-median clustering with outliers. In *Proc. of the ACM-SIAM Symposium on Discrete Algorithms*, 2008.

C. Croux and A. Ruiz-Gazen. A Fast Algorithm for Robust Principal Components Based on Projection Pursuit. In *Proc. in Computational Statistics*, 1996.

J.A. Cuesta-Albertos, A. Gordaliza, and C. Matrán. Trimmed k-means: an attempt to robustify quantizers. *The Annals of Statistics*, 25(2):553–576, 1997.

DARPA. DARPA Learning Applied to Ground Robots (LAGR). `http://www.nist.gov/el/isd/ms/lagr.cfm`, 2004. Accessed: 2015-03-30.

T. Eiter and H. Mannila. Computing Discrete Fréchet Distance. Technical Report CD-TR 94/64, Technische Universität Wien, 1994.

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 Knowledge Discovery and Data Mining*, 1996.

P. Felzenszwalb and D. Huttenlocher. Efficient Graph-Based Image Segmentation. *International Journal of Computer Vision*, 59(2):167–181, 2004.

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.

M. Fischler and R. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Communications of the ACM*, 24(6):381–395, 1981.

B. Frey and D. Dueck. Clustering by Passing Messages Between Data Points. *Science*, 315(5814):972–976, 2007.

C. Furtlehner, M. Sebag, and X. Zhang. Scaling Analysis of Affinity Propagation. *Physical Review E*, 81(6):066102, 2010.

L.A. Garcia-Escudero, A. Gordaliza, C. Matran, and A. Mayo-Iscar. A Review of Robust Clustering Methods. *Advances in Data Analysis Classification*, 4(2-3): 89–109, 2010.

A. Geiger, P. Lenz, and R. Urtasun. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

Y. Girdhar, P. Giguere, and G. Dudek. Autonomous Adaptive Exploration using Realtime Online Spatiotemporal Topic Modeling. *The International Journal of Robotics Research*, 33(4):645–657, 2014a.

Y. Girdhar, D. Whitney, and G. Dudek. Curiosity Based Exploration for Learning Terrain Models. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2014b.

I. Givoni and B. Frey. A Binary Variable Model for Affinity Propagation. *Neural Computation*, 21(6):1589–1600, 2009.

I. Givoni, C. Chung, and B. Frey. Hierarchical Affinity Propagation. In *Conference on Uncertainty in Artificial Intelligence*, 2011.

H. Hamacher and Z. Drezner. *Facility Location: Applications and Theory*. Springer, 2002.

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

C. Hennig. Dissolution point and isolation robustness: Robustness criteria for general cluster analysis methods. *Journal of Multivariate Analysis*, 99(6):1154–1176, 2008.

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*, 23(11-12):1005–1017, 2006.

P. Huber and E. Ronchetti. *Robust Statistics*. Wiley, 2008.

I. Jebari and D. Filliat. Color and Depth-Based Superpixels for Background and Object Segmentation. *Procedia Engineering*, 41:1307–1315, 2012.

E. Johnson and M. Hebert. Using Spin Images for Efficient Object Recognition in Cluttered 3D Scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(5):433–449, 1999.

R. Katz, J. Nieto, and E. Nebot. Unsupervised Classification of Dynamic Obstacles in Urban Environments. *Journal of Field Robotics*, 27(4):450–472, 2010.

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

K. Klasing, D. Althoff, D. Wollherr, and M. Buss. Comparison of Surface Normal Estimation Methods for Range Sensing Applications. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2009.

E. Knorr and R. Ng. A Unified Notion of Outliers: Properties and Computation. In *Proc. of the International Conference on Knowledge Discovery and Data Mining*, 1997.

A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet Classification with Deep Convolutional Neural Networks. In *Proc. of Advances in Neural Information Processing Systems*, 2012.

F. Kschischang, B. Frey, and H. Loeliger. Factor Graphs and the Sum-Product Algorithm. *IEEE Transactions on Information Theory*, 47(2):498–519, 2001.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

H. Lee, R. Grosse, R. Ranganath, and A. Ng. Convolutional Deep Belief Betworks for Scalable Unsupervised Learning of Hierarchical Representations. In *Proc. of the International Conference on Machine Learning*, 2009.

D. Lowe. Object Recognition from Local Scale-Invariant Features. In *Proc. of the IEEE Int. Conf. on Computer Vision (ICCV)*, 1999.

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.

F. Meyer and S. Beucher. Morphological Segmentation. *Journal of Visual Communication and Image Representation*, 1(1):21–46, 1990.

J. Modayil and B. Kuipers. Bootstrap Learning for Object Discovery. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2004.

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

D. Nistér, O. Naroditsky, and J. Bergen. Visual Odometry. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2004.

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*, 24(7):971–987, 2002.

E. Olson. Recognizing Places using Spectrally Clustered Local Matches. *Robotics and Autonomous Systems*, 57(12):1157–1172, 2009.

E. Olson, M. Walter, J. Leonard, and S. Teller. Single Cluster Graph Partitioning for Robotics Applications. In *Proc. of Robotics: Science and Systems (RSS)*, 2005.

L. Ott and F. Ramos. Real-Time Clustering for Long-Term Autonomy. In *Proc of the International Symposium of Experimental Robotics (ISER)*, 2012a.

L. Ott and F. Ramos. Unsupervised Incremental Learning for Long-Term Autonomy. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2012b.

L. Ott and F. Ramos. Multi-Sensor Clustering using Layered Affinity Propagation. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2013a.

L. Ott and F. Ramos. Unsupervised Online Learning for Long-Term Autonomy. *The International Journal of Robotics Research*, 32(14):1724–1741, 2013b.

L. Ott, L. Pang, F. Ramos, and S. Chawla. On Integrated Clustering and Outlier Detection. In *Proc. of Advances in Neural Information Processing Systems*, 2014.

S. Papadimitriou, H. Kitagawa, P. Gibbons, and C. Faloutsos. LOCI: Fast Outlier Detection Using the Local Correlation Integral. In *Int. Conf. on Data Engineering*, 2003.

R. Paul and P. Newman. Self Help: Seeking Out Perplexing Images for Ever Improving Navigation. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2011.

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

D. Pelleg and A. Moore. X-means: Extending K-means with Efficient Estimation of the Number of Clusters. In *Proc. of the International Conference on Machine Learning*, 2000.

A. Quadros, J. Underwood, and B. Douillard. An Occlusion-Aware Feature for Range Images. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2012.

S. Ramaswamy, R. Rastogi, and K. Shim. Efficient Algorithms for Mining Outliers from Large Data Sets. In *Int. Conf. on Management of Data*, 2000.

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.

P. Rousseeuw. Least Median of Squares Regression. *Journal of the American statistical association*, 79(388):871–880, 1984.

P. Rousseeuw and K. Driessen. A fast algorithm for the minimum covariance determinant estimator. *Technometrics*, 41(3):212–223, 1999.

J. Schoenberg, A. Nathan, and M. Campbell. Segmentation of Dense Range Information in Complex Urban Scenes. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2010.

A. Schrijver. *Theory of Linear and Integer Programming*. John Wiley & Sons, 1998.

J. Shi and J. Malik. Normalized Cuts and Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

M. Smith, I. Baldwin, W. Churchill, R. Paul, and Newman P. The New College Vision and Laser Data Set. *The International Journal of Robotics Research*, 28(5):595–599, 2009.

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.

B. Steder, R.B. Rusu, K. Konolige, and W. Burgard. Point Feature Extraction on 3D Range Scans Taking into Account Object Boundaries. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2011.

J. Sun, J. Rehg, and A. Bobick. Learning for Ground Robot Navigation with Autonomous Data Collection. Technical Report GIT-GVU-05-29, Georgia Institute of Technology, 2005.

R. Triebel, R. Paul, D. Rus, and P. Newman. Parsing Outdoor Scenes from Streamed 3D Laser Data Using Online Clustering and Incremental Belief Updates. In *AAAI Conference on Artificial Intelligence*, 2012.

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

L. Vincent and P. Soille. Watersheds in Digital Spaces: An Efficient Algorithm Based on Immersion Simulations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(6):583–598, 1991.

U. von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17 (4):395–416, 2007.

C. Wang, J. Lai, C. Suen, and J. Zhu. Multi-Exemplar Affinity Propagation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(9):2223–2237, 2013.

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

Y. Weiss, C. Yanover, and T. Meltzer. MAP Estimation, Linear Programming and Belief Propagation with Convex Free Energies. In *Conference on Uncertainty in Artificial Intelligence*, 2007.

J. Wright, A. Ganesh, S. Rao, Y. Peng, and Y. Ma. Robust Principal Component Analysis: Exact Recovery of Corrupted Low-Rank Matrices by Convex Optimization. In *Proc. of Advances in Neural Information Processing Systems*, 2009.

D. Xia, F. Wu, X. Zhang, and Y. Zhuang. Local and global approaches of affinity propagation clustering for large scale data. *Journal of Zhejiang University SCIENCE A*, 9(10):1373–1381, 2008.

J. Xiao, J. Wang, P. Tan, and L. Quan. Joint Affinity Propagation for Multiple View Segmentation. In *Proc. of the IEEE Int. Conf. on Computer Vision (ICCV)*, 2007.

L. Zelnik-Manor and P. Perona. Self-Tuning Spectral Clustering. In *Proc. of Advances in Neural Information Processing Systems*, 2004.

D. Zhang, C. Lin, S. Chang, and J. Smith. Semantic Video Clustering Across Sources using Bipartite Spectral Clustering. In *IEEE International Conference on Multimedia and Expo*, 2004.

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.