

Master Thesis

Towards lifelong mapping in pointclouds

Panagiotis Chatzichristodoulou

Master Thesis DKE 09-16

Thesis submitted in partial fulfillment
of the requirements for the degree of Master of Science
of Artificial Intelligence at the Department of Knowledge
Engineering of the Maastricht University

Thesis Committee:

University of Maastricht

Rico Mockel, Kurt Driessens

Dobots B.V.

Anne Van Rossum

Maastricht University

Faculty of Humanities and Sciences

Department of Knowledge Engineering

Master Artificial Intelligence

July 6, 2015

Abstract

Long term mapping is the natural conceptual extension to existing mapping methods that are focused on mapping static environments. Mapping an area for a large amount of time is a problem that if tackled efficiently will be a large step towards highly autonomous robots. As discussed in the literature lifelong mapping consists of two major subproblems. A compression, and a dynamic environment problem. The thesis discusses the application of non parametric Bayesian methods and how such tools can be directed to tackle the clustering subproblem of lifelong mapping methods. A novel method of point cloud representation and matching is presented and its results are applied to an EKF SLAM algorithm; its strengths and weaknesses are analyzed as well as the directions in which the method could be extended to be a complete solution to both the dynamic as well as the compression problem of long term slam.

CONTENTS

1	Introduction	4
2	Literature review	5
2.1	Object based SLAM	5
2.2	Point Cloud Object segmentation	6
2.3	Non Parametric Bayesian methods	6
2.4	Correspondence	7
3	Theory background	8
3.1	Generalized Polya's Urn	8
4	Model definition	9
4.1	General pipeline	9
4.2	The data distribution	12
4.3	Sequential monte carlo sampler	14
4.3.1	Gibbs updates	14
4.3.2	Weight updates	16
4.4	Decision Layer	16
4.5	Complexity	17
4.6	Landmark size	18
5	Results	18
5.1	Simple datasets	18
5.2	The decision layer	20
5.3	EKF-SLAM experiments	23
5.3.1	Speed	23
5.3.2	Memory requirements	24
6	Conclusion and future work	24
7	Discussion	24

LIST OF FIGURES

1	Generalised Polya Urn as presented in CRP.	9
2	General landmark update pipeline	10
3	Point cloud modification pipeline.	11
4	Exponential trend in KL distances.	12
5	Initial data along with the distributions inferred	19
6	More complicated color distributions	21
7	The colour and position boudnary is displayed in these pictures	22
8	caption	25

1. INTRODUCTION

Simultaneous localization and mapping is one of the fundamental problems of autonomous systems[1]. In order for a robot to be considered truly autonomous, it must have the ability to enter an area and infer its structure. To that direction, a lot of effort has been put in algorithms that are able to map static environments. With solutions like EKF-SLAM[2], FastSlam[40] and GraphSLAM[3] robots are now able to efficiently map static environments. The logical extension to methods that can map static environments is methods that remove this restriction. The idea of lifelong robot learning is not new and has been introduced as a general concept to the literature by Sebastian Thrun [5]. Konolige et al[6] specifically focus on lifelong learning in mapping and the utility such methods would have. In the PhD thesis of Walcott [7] long term mapping methods are decomposed to 4 basic subproblems:

- A method that is able to continuously incorporate new information.
- A method that addresses the problem of tractability for growing DPG
- A method that detects changes and update the map online
- Representation of the environment should include the history of the map as changes occur with the passage of time.

The first two problems can be thought of as compression problems as the map increases over time whereas the latter ones can be thought of as dynamic environment problems. Approaches of tackling those problems vary according to the sensors a robot is using to perform the mapping. In this project the focus will be directed towards methods that use RGBD devices like Microsoft's Kinect to perform SLAM.

Since its introduction in 2010 Microsoft Kinect[25] has revolutionized RGBD devices with its low price range and high quality sensors. It came as no surprise that research in point clouds, the cloud representation system of Kinect, has increased since then. Many libraries that enable the user to perform tasks from feature extraction to plane segmentation[12] in pointclouds are currently available. In the field of robotics, many teams are using the Kinect sensors to perform simultaneous localization and mapping[13]. The goal of this thesis is to introduce a novel approach to tackle the compression problem of long term mapping methods that use the Kinect devices while using Bayesian non parametric methods as the base of its solution.

Dirichlet processes and Dirichlet process mixture models [26] are the cornerstone of Bayesian non parametric statistics. The strength of those models lies in the fact that they allow the model's mixture components to grow as much as needed so as to best fit the data. The dynamic number of components in combination with many different prior choices lead to very flexible models that can be used in a very large area of applications from topic modeling[43] to speaker diarization[45].

The main motivation behind this project is to use such methods as a means of compressing the information provided by the environment. In that direction, finding a way to robustly cluster a point cloud into "chunks" of structure seems a reasonable starting point. This leads to the direction of object based SLAM, which is a domain where objects are used as reference points to perform the mapping.

In this paper, an EKF SLAM method that uses point clouds as input will be implemented. More specifically, a landmark detection layer will be incorporated; its results will then be empirically

tested as well as its ability to optimally compress the data of a point cloud. Its strengths and weaknesses will be presented and analysed; furthermore, directions on how the method could be extended to tackle all four subproblems of Walcott's thesis will be given in the discussion.

The rest of the paper is structured as follows. Section 2 will present relevant literature review, Section 3 will introduce the theories behind the model, Section 4 will define the model, Section 5 will show experimental results of the method. Finally, Section 6 will end up with a conclusion and future directions in which the method could be improved.

2. LITERATURE REVIEW

Related research will be focused on 4 general sub fields of related literature.

- Object based SLAM or semantic slam
- Point Cloud Object segmentation
- Non-parametric clustering methods
- The correspondence problem in SLAM

Since point clouds are used as input and from this input object representations must be extracted, methods that use objects to perform SLAM are then needed. Object based SLAM methods are therefore important given to the nature of the input. The second part of the literature research is focused on point cloud representations and is specifically directed towards the specific meta-features that need to be taken into account so that the reduced point cloud representation gives sufficiently enough information of the cloud. The third part of the research is focused on non-parametric Bayesian methods and the clustering tools they provide. Such tools are important as they can be used to provide new approaches to object segmentation in a point cloud. Finally, research is focused on the correspondence problem in SLAM. As one of the fundamental problems that need to be solved in order to have robust SLAM algorithms, it is imperative the correspondence problem be solved efficiently. In that extent and due to the unique representation of our objects, a novel approach that takes elements from other techniques in the correspondence between objects in a SLAM problem is presented.

2.1. Object based SLAM

Object based SLAM or semantic slam methods proposed in the literature focus at domain specific solutions and particular problems. Salas-Moreno et al [14] define a method of performing object based slam for specific classes objects. The objects are identified by camera that is on top of the robot. By having a model of pre-trained objects, SLAM can be performed on environments the robot knows what objects to expect. The disadvantage of that method is that object models have to be well defined and there is a small number of such objects. Castle et al. use object recognition to perform object based SLAM with the use of a hand-held cameras. Selvatici et al [15] use a similar approach while exploiting structural information such as object height and position within the room. That way a couch that is a large object situated in floor level is easier to be recognized. Choudhary et al. [17] use point clouds and an object database to match objects currently seen

with known objects within their database. They use omnimaper [27] as their mapping method and as a representation a combination of the downsampled voxel grids with additional normal and curvature information. Finally, all their operations are done in the non-planar components of the point cloud. Jensfelt et al [15] present an object based approach to SLAM where the robot can manipulate the objects of the map. They use camera pictures as input and receptive Field Histogram as the method to abstract the camera input and extract features for their object matching algorithm. Their approach is proposed as a solution to a service robot scenario. MonoSLAM [19] introduces a method of performing slam using a monocular camera.

What all those methods have in common is that they approach the problem of object based slam as a classification task. Objects need to be semantically understood before they are processed. The approach introduced in this paper considers the environment to be a collection of chunks. So having specific enough environment descriptors should lead to the robot being able to operate in an environment without having to explicitly label every object it encounters. This would remove the need of having to classify objects but would also increase the time it takes to extract features from the environment as features are the base of the unsupervised object discovery. Seongyong Koo et al. [20] introduce a method of unsupervised object individuation from RGB-D image sequences. They cluster their initial cloud into candidate objects using Euclidian clustering and proceed to extract features like the Euclidian distance(L2) and the Kullback-Leibler distance between point cloud objects. They use IMFT to solve their tracking problem.

2.2. Point Cloud Object segmentation

Research towards object segmentation in point clouds is focusing on calculating meta information regarding the points and applying a heuristic function to calculate if the points could belong in the same segment of the cloud. Trevor et al. [29] take positional information, Euclidean distances and the normal of points to as input to their heuristic and output segments that are part of the same object. PCL library [12] introduces methods like Euclidean clustering and conditional Euclidean clustering that use a number of heuristics that take normal as well as curvature information to extract segments in the point cloud that represent objects. Furthermore, a there is a lot of research on segmentation of point clouds in scenes, the emphasis is usually on extracting geometric primitives [30], [31] using cues like normals and curvature. Rabbani et al [23] introduce a new method of object segmentation using KNN as their base algorithm. They also present a very informative literature review along with the strengths and weaknesses of existing methods. Finally Triebel et al. [32] introduce a general clustering framework that does not rely on plane segmentation. Instead of segmenting the plane by using classical approaches like RANSAC or MLASAC they introduce a framework where they make no assumptions regarding plane data.

2.3. Non Parametric Bayesian methods

Dirichlet processes and Dirichlet process mixture models are the cornerstone of Bayesian statistics. In this project the focus was directed towards the clustering methods that are being introduced by those tools. Radford M. Neal [38] with his paper regarding MCMC methods for Dirichlet process mixture models made the definitive step towards Dirichlet process mixture models(DPMM's) receiving a lot of attention. Since then, a variety of approaches for inference on such models has

been introduced with MCMC methods, and variational inference methods being two prominent such approaches. Variational inference for DPMM's, introduced by Jordan et al. [39] introduces deterministic tools to perform inference and approximate the posterior distribution and marginals of a dataset. Both methods have strengths and weaknesses and many tools have been established by using the two approaches as their base. Blei et al. [43] introduced LDA as a method to perform topic modelling. Teh et al [41] introduce a hierarchy on the inference process by introducing the Hierarchical Dirichlet process. Particle filter approaches have also been established. Doucet et al. [42] introduce Sequential Monte Carlo as a fast way to approximate inference. Dirichlet process mixture models are a very active research field and covering it is beyond the scope of this report. In this project SMC samplers were used due to their robustness as well as their inherent extensiveness. A detailed description on the mechanisms of sequential samplers will be given in the theory section.

2.4. Correspondence

In its general definition, the correspondence problem refers to the problem of ascertaining which parts of one image correspond to which parts of another image, where differences are due to movement of the camera, the elapse of time, and/or movement of objects in the photos. Under the semantic SLAM context, it refers to the problem of identifying objects as ones that have been encountered before during the mapping process. Towards that direction Cree et al. [34] create a histogram of line segments of each landmark and compute their root mean square error. They then proceed to calculate their RGB signature to calculate the distance between different landmarks. Low et al. [35] match Scale Invariant Feature Transform (SIFT) features, an approach which transforms image data into scale-invariant coordinates relative to local features. Lamon et al [36] store a database of fingerprints which indicate the location in the robot's environment. The features are ordered and stored at a database as they appear in the robot's immediate surroundings. A new fingerprint is computed for each new view and matched against existing ones. Finally, in Seghal et al. [37] an extension of SIFT descriptors to 3D data and point clouds is given.

The method presented in this paper receives an input similar to [20] as parts of the point cloud are being clustered a mixture of distributions. The features that are used for the cloud representation are an extension of the features presented in [33] with the addition of angle information extracted in the points of the cloud. The novelty lies in its probabilistic mechanism as the clustering is done by the SMC sampler in the augmented feature space. Using distributions as a means of representing objects within the point cloud is a form of compression as objects are represented by a distribution which is smaller in size and easier to maintain and expand. The matching then is done by calculating the divergences between distributions. The pipeline is presented in detail in the model definition section .

3. THEORY BACKGROUND

3.1. Generalized Polya's Urn

Dirichlet process priors have been widely used in the literature as non parametric Bayesian tools to estimate the number of clusters in the data [46]. Dependent dirichlet processes extend those priors by allowing the clusters in the data to vary with some covariance over time. Dependent Dirichlet processes (DDP) remove the restriction of exchangeable data and introduce dependencies which can be temporal, positional etc. The DDPs are a natural extension of the DP's in domains where data cannot be considered exchangeable and have dependencies that cannot be disregarded. They were introduced by MacEachern [44] and have been widely used since. The main motivation behind using such methods is the immediate extension they provide to dynamic environments.

A DDP also known as Generalized Polya Urn [47] and has the property of randomly deleting partitions of clusters on every iteration. That way, it can cope with the variance of the data. In the current project, the n _{*t*}th datapoint at time $t, x_{t,n}$ has an assignment $c_{t,n}$ at cluster $k \in \{1, 2, \dots, K\}$. The size of cluster k at time t is defined as s_t^k . The GPU of this model at time t can now be defined as:

Algorithm 1 GPU

```

1: procedure GPU(pointCloud, t)
2:   for  $k = 1, \dots, K_{t-1, N_{t-1}}$  do
3:     Draw  $\Delta s_{t-1}^k \sim \text{Binom}(s_{t-1, N_{t-1}}^k, \rho)$  ▷ Number of elements to delete
4:     Set  $s_{t,0}^k = s_{t-1, N_{t-1}}^k - \Delta s_{t-1}^k$ 
5:   end for
6:   for  $n = 1, \dots, N_t$  do
7:     Draw  $c_{t,n} \sim \text{Cat}(\frac{s_{t,n-1}^1}{\alpha + \sum_k s_{t,n-1}^k}, \frac{s_{t,n-1}^{K_{t,n-1}}}{\alpha + \sum_k s_{t,n-1}^k}, \frac{\alpha}{\alpha + \sum_k s_{t,n-1}^k})$ 
8:     If  $c_{t,n} \leq K_{t,n-1}$  set :  $s_{t,n}^{c_{t,n}} = s_{t,n-1}^{c_{t,n}} + 1, K_{t,n} = K_{t,n-1}$ 
9:     If  $c_{t,n} > K_{t,n-1}$  set :  $s_{t,n}^{c_{t,n}} = 1, K_{t,n} = K_{t,n-1} + 1$ 
10:   end for
11: end procedure

```

Where Cat is a categorical distribution, Bin is the binomial distribution, α is the DP concentration parameter and ρ is the deletion parameter of the GPU. This Generative Polya Urn distribution also has the shorthand notation GPU(α, ρ)

This process can be thought of in the terms of a general Chinese restaurant process as shown in Fig. 1. At time t (a), suppose there are customers seating at several tables in the restaurant. Each customer has to decide if he/she will remain at table with probability p or leave the restaurant with probability $1 - p$. Once all the customers make their decisions they leave the restaurant or remain seated (b). Each table occupied is moved according to the number of customers still seating in that table (c). A new customer then enters the table and either chooses to sit on one of the existing tables (e) or choose a new with probability proportional to the strength parameter α of the model (f).

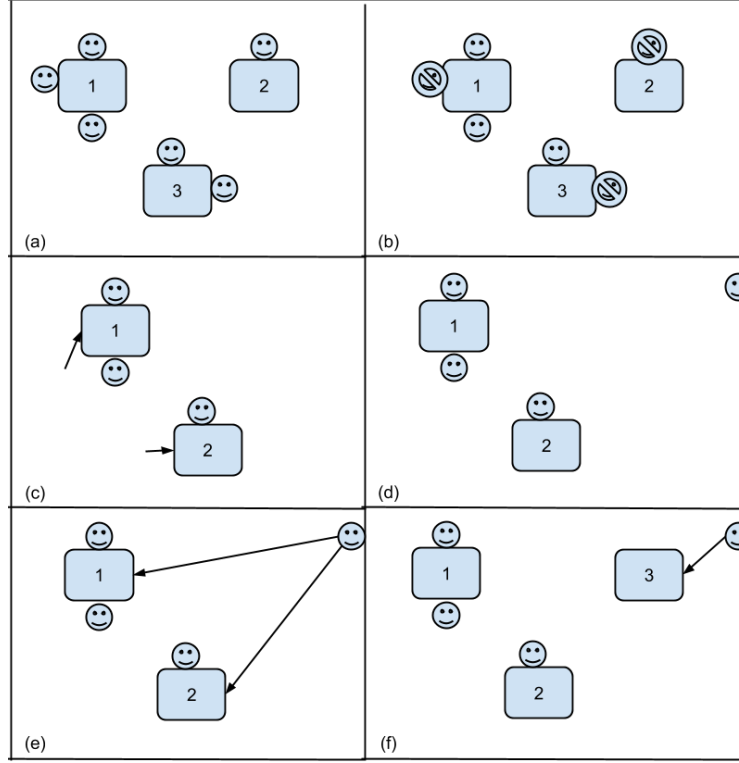


Figure 1: Generalised Polya Urn as presented in CRP.

4. MODEL DEFINITION

4.1. General pipeline

Now that the data distribution and sampler are defined, the model that takes as input point clouds from a Kinect sensor mounted on a robot and outputs the most probable landmarks given the input and the previous landmarks can be presented. Fig. 2 depicts the general workflow of what happens during the observation step of SLAM. The SLAM node– requires landmark information, and the landmark pipeline outputs the most probable landmarks given the current database of landmarks and the current observations. If needed landmarks added to the database and the landmark ids are given back to the SLAM node.

The model introduces a Pipeline in the layer of landmark detection of the EKF-SLAM algorithm. All the operations from feature extraction to clustering and object matching will be performed at that layer. More specifically the pseudo-code for the method is the following:

This top level description has a lot of implied steps so a line by line description will be provided:

Method input: The method takes as input a point cloud. The post transformation comment has to do with the fact that the cloud expected is the one after all the frame transformations are

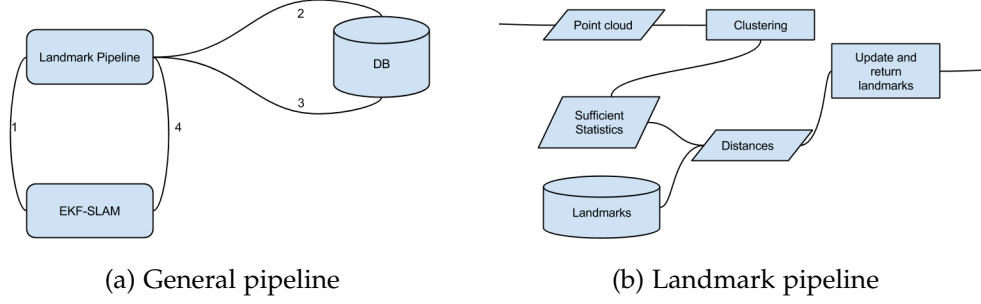


Figure 2: General landmark update pipeline

Algorithm 2 Landmark Layer

```

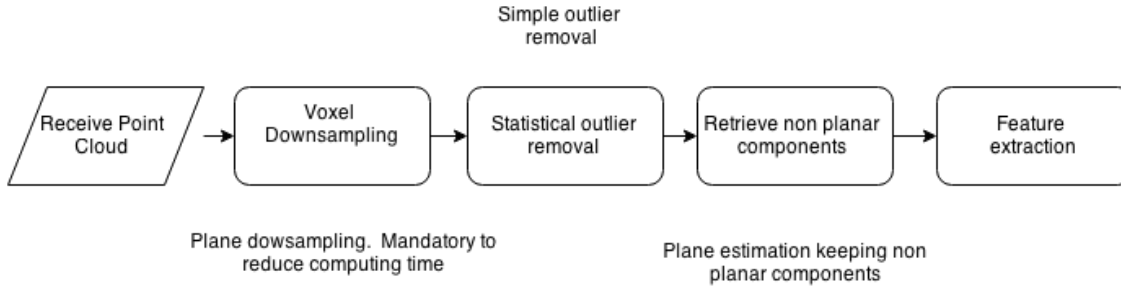
1: procedure GETLANDMARKIDS(pointCloud, timepoint, existingLandmarks) ▷ Post transformation
2:   initialize(landMarkIds) ▷ Initialize empty list
3:   pointCloudReduced ← extractMetaFeatures(pointCloud) ▷ Cloud preprocessing
4:   features ← extractMetaFeatures(pointCloudReduced)
5:   landmarks ← cluster(features) ▷ Cluster using features
6:   for landmarks as landmark do
7:     (similarity, landId) ← calcBestSim(landmark, existingLandmarks)
8:     if similarity > threshold then
9:       addLandmarks(landMarkIds, landId) ▷ Return known landmark
10:    else
11:      newLandID ← addLandmarkDB(landmarkDB, landmark) ▷ Add get new id
12:      addLandmarks(newLandID) ▷ Add landmark
13:    end if
14:  end for
15:  return landMarkIds ▷ Return added landmark
16: end procedure

```

done in the tf layer of the robot.

Lines 3-4: Feature extraction is done through the pcl [12] library. An initial point cloud reduction is mandatory to increase the speed of the process. A voxel grid is used to reduce the dataset size. A leaf size of approximately 4cm produces a good tradeoff between precision and speed. The object representation approach is similar to [17]. Instead of using the CSHOT descriptor, pcl's fpfh [22] is being used. Fast point feature histogram(fpfh) represents an angular signature between a point and its neighbors. In that way we end up with a 23 dimensional angular signature of information between a point and its neighbors. Since the information are given in the form of a histogram, classical statistic solutions regarding the distances between angular signatures can be taken into account. In our solution EMD, Hellinger, and KL divergence are being computed in the pipeline. Color information is being encoded with an approach similar to [33]. The colour spectrum is discretized and what is extracted is the count of different colour signatures between a point and its k nearest neighbors. Finally positional information is also given as input to the algorithm. The pipeline is presented in figure Fig. 3. What the algorithm outputs is then a vector of $\mathbf{x} = (x_s, x_c, x_a)$ where s represents a 3x1 vector of space information, c a 27x1 vector of colour information and a angular information whose dimensionality depends on the distances computed (in our case 3x1). It must be noted that pcl offers a multi threading operation on the point feature histogram extraction. This feature is important since it greatly reduces (6-8 times) the speed of an operation applied on every point of the cloud and that makes the histogram extraction significantly faster.

Figure 3: Point cloud modification pipeline.



Lines 5: The clustering takes place in this line. The input of the method is the feature vector for every data point which is calculated in the previous steps. An SMC sampler is used as was presented in the theory section.

Lines 6-12: The correspondence of previously seen landmarks to current observations is computed in lines 6-12. Since the landmarks are distributions, statistical distances can be taken to perform the matching. For every observation, its distances with all the stored landmarks are calculated. The calcBestSim function takes as input the observation and all the previously seen landmarks and returns highest similarity match with the landmarks already in the database as well as its landmark id. If the similarity is high enough, correspondence is performed and

the landmark is added to the landmark list to be send for update in the EKF, otherwise a new landmark is added and its ID is then added to the list.

Lines 15: The algorithm returns the list of the most probable landmarks the robot encounters in this current time.

Now that the general pipeline is defined, specific parts of the method will be defined in more detail

4.2. The data distribution

The point cloud consists of many points hence a data unit will be considered a single point. Each point x in the input consists of a tuple $x = (x^s, x^a, x^c)$ where superscript s represents spatial information, a angle information, and c colour information. The method those features are extracted is explained in lines 3 and 4 in the general pipeline. For the purpose of this project each part of the cloud was represented by vector of length 32 with the first three elements representing the space information, elements 4-6 angle information, and the rest colour information.

The object model is a mixture of distributions over the data with each object beign modeled as $D(\theta_t^k)$ where θ represents the parameters of object k at time t . More specifically, each set x with n datapoints at time t is distributed as:

$$x_{t,n} \sim D(\theta_t^k) = \text{Normal}(x_{t,n}^s | \mu_t, \Sigma_t) \text{Mult}(x_{t,n}^c | \delta_t) \text{Exp}(x_{t,n}^a | \lambda_t)$$

Where Normal is a three dimensional Gaussian distribution with mean μ and covariance Σ representing the positional distribution of the data; Mult is a Categorical multinomial distribution with parameter vector δ representing weigths the colour distribution and Exp is an exponential distribution with shape parameter λ representing the angle distribution of the data within the cluster. The exponential distribution was chosen to model angular information after empiric evaluation showed that it would be a good fit for the angle signature distribution of the data. A typical angle signature distribution is shown in. 4. For this graph data points from various point clouds where taken and passed through the pipeline analysed in lines 3-4. The data were then plotted, and the figure shows what the aggregated angle distribution between neighbor datapoints is. Since there is a clear exponential trend, the exponential distribution was chosen to model it. A gamma distribution is also a nice fit for the data, but due to time constrains, only one modeling distribution was explored.

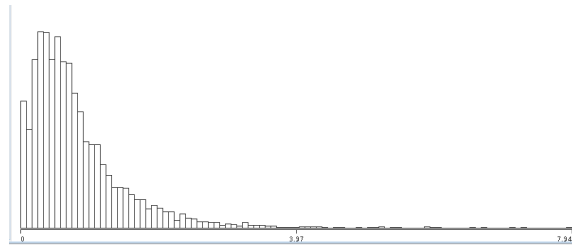


Figure 4: Exponential trend in KL distances.

Now that the distribution of the objects is defined, the progression of the sufficient statistics at time t given $t - 1$ given by:

$$\theta_t^k | \theta_{t-1}^k \sim \begin{cases} T(\theta_{t-1}^k) & \text{if } k \leq K_{t-1} \\ G_0 & \text{if } k > K_{t-1}. \end{cases}$$

Where T represents the transition kernel of the data given the previous state in the model. The case $k > K_{t-1}$ represents the creation of a new cluster and G_0 is the base distribution of the DDP. In our case, the conjugate priors of the distributions of the data were chosen to model the base distribution. Therefore, G_0 is defined as:

$$G_0(\theta_t^k) = NiW(\mu_t^k, \Sigma_t^k | \kappa_0, \mu_0, \nu_0, \Lambda_0) Dir(\delta_t^k | q_0) Gam(\lambda_t^k | \alpha_0, \beta_0)$$

Where NiW is a Normal inverse Wishart distribution, Dir denotes a Dirichlet distribution, and Gam the Gamma distribution. $\kappa_0, \mu_0, \nu_0, \Lambda_0, q_0, \alpha_0$ and β_0 are predefined parameters of the model. The generative process for the Dependent Dirichlet mixture model can be written for each timestep t as:

-
1. Draw $c_t \sim GPU(\alpha, \rho)$
 2. $\forall k$ draw: $\theta_t^k | \theta_{t-1}^k \sim \begin{cases} T(\theta_{t-1}^k) & \text{if } k \leq K_{t-1} \\ G_0 & \text{if } k > K_{t-1}. \end{cases}$
 3. \forall point n draw $x_{t,n} \sim F(\theta_t^{c_t, n})$
-

Given the theory in [47], the transition Kernel must satisfy:

$$\int G_0(\theta_k) T(\theta_t^k | \theta_{t-1}^k) d\theta_{t-1}^k = G_0(\theta_k)$$

The equation means that the invariant distribution must equal its base distribution. A typical way of meeting this restriction and forcing the sampler to converge to the original target density[49] is to introduce a set of M auxiliary variables \mathbf{z} such that:

$$P(\theta_t^k | \theta_{t-1}^k) = \int P(\theta_t^k | z_t^k) P(z_t^k | \theta_{t-1}^k) dz_t^k$$

The transition kernel of the model can now be sampled by using the following formula $\theta_t^k \sim T(\theta_{t-1}^k) = T_2 \circ T_1(\theta_{t-1}^k)$ where:

$$\begin{aligned} z_t^k &\sim T_1(\theta_{t-1}^k) \\ &= Normal(\mu_{t-1}, \Sigma_{t-1}) Mult(\delta_{t-1}) Exp(\lambda_{t-1}) \end{aligned} \tag{1}$$

$$\begin{aligned} \mu_t, \Sigma_t, \delta_t, \lambda_t &\sim T_2(z_t^k) \\ &= NiW(\kappa_0, \mu_0, \nu_0, \Lambda_0) Dir(q_0) Gam(\alpha_0, \beta_0) \end{aligned} \tag{2}$$

where $\mu_t, \Sigma_t, \delta_t, \lambda_t$ are posterior parameters given the auxiliary variables \mathbf{z} .

4.3. Sequential monte carlo sampler

Sequential monte carlo samplers for Dirichlet process mixture models were introduced by Doucet et al [50] and serve as fast alternative to MCMC and VI methods of performing posterior inference. SMC samplers have known strengths and weaknesses and are a good fit for the problem at hand, as their main theoretical disadvantage which is the particle degradation cannot occur in a time horizon of 1 that the sampler is being used. We can now define the SMC sampler that will be used to perform inference on our model as follows:

Algorithm 3 SMC for DDPM

```

1: Input: Points  $\{x_{1,1:N_t}, \dots, x_{T,1:N_t}\}$  with extracted features
2: Output: Clusters representing of the data
3: for  $t = 1, \dots, T$  do
4:   for  $l = 1, \dots, L$  do
5:     for  $iter = 1, \dots, S$  do
6:       Sample  $(c_t)^{(l)} \sim Q_1$ 
7:       Sample  $(\theta^k) \sim Q_2$ 
8:     end for
9:   end for
10:  for  $k = 1, \dots, K$  do
11:    Sample  $\Delta s_{t-1}^k \sim \text{Binom}((s_{t-1, N_{t-1}}^k)^{(l)}, \rho)$ 
12:    Set  $s_{t,0}^k = s_{t-1, N_{t-1}}^k - \Delta s_{t-1}^k$ 
13:    Sample  $((z_{t+1}^k)^{(l)}) \sim T_1((\theta_t^k)^{(l)})$ 
14:  end for
15:  compute particle weights  $w_t^l$ 
16: end for
17: Normalize and resample weights
    
```

4.3.1 Gibbs updates

The proposal distribution Q_1 is the probability of an assignment $c_{t,n}$ given cluster sizes, parameters and concentration α . Formally Q_1 can be written as:

$$Q_1(c_{t,n} | s_{t,n}^k, \theta_t^k, \alpha) \propto \text{Cat}(s_{t,n}^1, \dots, s_{t,n}^K, \alpha) \times \begin{cases} F(x_{t,n} | \theta_t^{c_t}) & \text{if } k \leq K_{t-1} \\ \int P(x_{t,n} | \theta_t) G_0(\theta) d\theta & \text{if } k > K_{t-1}. \end{cases} \quad (3)$$

Where $c_{t,n}$ represents cluster c of point n at time t , s represents cluster sizes. The integral represents the posterior predictive distribution of the cluster times the base distribution with the parameters integrated out. A review of the literature helps understand how the posterior predictive formula

is derived. More specifically, the analytic expression of the integral is:

$$\begin{aligned}
 \int P(x_{t,n}|\theta_t)G_0(\theta)d\theta &= \int Normal(x_{t,n}^s|\mu_t, \Sigma_t) Mult(x_{t,n}^c|\delta_t) Exp(x_{t,n}^a|\lambda_t) \times \\
 &\quad NiW(\mu_t, \Sigma_t|\kappa_0, \mu_0, \nu_0, \Lambda_0) Dir(\delta_t|q_0) Gam(\lambda_t|\alpha_0, \beta_0) d\theta \\
 &= \int Normal(x_{t,n}^s|\mu_t, \Sigma_t) \times NiW(\mu_t, \Sigma_t|\kappa_0, \mu_0, \nu_0, \Lambda_0) \\
 &\quad Mult(x_{t,n}^c|\delta_t) \times Dir(\delta_t|q_0) \\
 &\quad Exp(x_{t,n}^a|\lambda_t) \times Gam(\lambda_t|\alpha_0, \beta_0) d\theta \tag{4} \\
 &= t_{\nu_0-1}(x_{t,n}^s|\mu_0, \frac{\Lambda_0(\kappa_0+1)}{\kappa_0(\nu_0-1)}) \times \prod_{j=1}^V \frac{\Gamma(x_{t,n}^c)}{\Gamma(q_0)} \times \\
 &\quad \frac{\Gamma(\sum_{j=1}^V q_0)}{\Gamma(\sum_{j=1}^V x_{t,n}^c)} \times Lomax(\alpha_0 + s_{t,n}^c, \beta_0 \sum_{j=1}^V x_{t,n}^c)
 \end{aligned}$$

Where t represents student's t-distribution with ν degrees of freedom, Lomax represents Lomax distribution with shape and scale, α and β respectively and the rest represent a Dirichlet-Multinomial(aka DirMul) distribution. The formulas of the posterior predictive distributions can be found in the literature with [48] being a good example.

The conjugacy of the base and prior distribution allows for an easy sampling formula for proposal distribution Q_2 which is of the form:

$$\begin{aligned}
 Q_2(\theta_t^k|\theta_{t-1}^k, x_t^k, z_t^k) &\propto F(x_t^k|\theta_k) \times T_2(\theta_t^k|z_t^k) \\
 &= NiW(\mu_t^k, \Sigma_t^k|\kappa_n, \mu_n, \nu_n, \Lambda_n) Dir(\delta_t^k|q_n) Gam(\lambda_t^k|\alpha_n, \beta_n) \tag{5}
 \end{aligned}$$

With:

$$\begin{aligned}
 \kappa_n &= \kappa_0 + N \\
 \nu_n &= \nu_0 + N \\
 \mu_n &= \frac{\kappa_0}{\kappa_0 + N} \mu_0 + \frac{N}{\kappa_0 + N} \bar{x}^s \\
 \Lambda_n &= \Lambda_0 + s_x^s \\
 q_n &= q_0 + \sum_n x_i^c \\
 \alpha_n &= \alpha_0 + N \\
 \beta_n &= \beta_0 + \sum_n x_i^a \tag{6}
 \end{aligned}$$

Where \bar{x} defines the sample mean for the elements assigned at cluster c , s_x the sample variance and N denotes the number of observations. The formulas for the updates can be found at the literature of conjugate priors with [52] being a good example.

4.3.2 Weight updates

The only thing left to explicitly define the sampler in its whole is the weight update step. More specifically, on every time step t the weight of particle l is calculated as:

$$w_t^{(l)} = \frac{P(c_t^{(l)}, \theta_t^{(l)}, x_t | \theta_{t-1})}{P(c_t^{(l)}, \theta_t^{(l)} | \theta_{t-1})} \quad (7)$$

Using bayes rule, the numerator can be written as:

$$P(x_t, |c_t^{(l)}, \theta_t^{(l)} | \theta_{t-1}) \times P(c_t^{(l)}, \theta_t^{(l)} | \theta_{t-1}) \quad (8)$$

Which can be calculated using equations Q_2 and Q_1 for the first and second part respectively. After the particle weights are normalized particles are drawn with probability proportional to their weights.

4.4. Decision Layer

Once the sampler outputs the clusters of the point cloud the output can be passed to the decision layer. The question that the decision layer answers is: "How does the data/clusters I currently observe relate to the data I have previously seen in the past?". The decision of how an observation relates to previously encountered data has to be made and this decision is taken in the decision layer of the algorithm. To do that, distance measures must be defined between the observations and the landmarks. Distances between distributions are called divergences and a large amount of literature on divergences exists.

Since our data are modeled as coming from a convolution of 3 distributions, specific divergences for each part were considered. More specifically let l be the distribution of the landmark and o the distribution of the observation. l can be decomposed into 3 parts: l_G, l_C, l_E where G, C and E stand for Gaussian, Categorical and Exponential respectively. The observation distribution can be decomposed similarly. With that notation the distances between those distributions can be defined. For each individual landmark distribution l and observation distribution o the distances computed where the following:

- Gaussian
 - Wasserstein(l_G, o_G);
 - KL(l_G, o_G);
- Exponential
 - SquaredHellinger(l_E, o_E);
 - KL(l_E, o_E);
- Categorical
 - KL(l_C, o_C);

The decision boundary of an observation being a landmark or not was chosen by empiric evaluation of the landmarks. It is of course possible to learn the optimal decision boundary but due to time restrictions a simpler decision making approach was chosen instead.

4.5. Complexity

The complexity can be decomposed into three major parts. The cloud downsampling, the clustering and the decision making.

$$O(\text{total}) = O(\text{filter}) + O(\text{cluster}) + O(\text{decision})$$

Downsampling: The complexity of the cloud downsampling pipeline can be decomposed to the one of its components. This means that the decomposed complexity is defined as follows:

$$O(\text{filter}) = O(\text{Vox Downsampling} + \text{Stat Removal} + \text{RANSAC} + \text{FPFH} + \text{Color estimation})$$

Voxel downsampling searches for neighbors within a distance defined by the user and keeps an average value that equally represents the cloud. Since the operation involves searching for neighbors of a point, and since search operations take $O(\log n)$ time where N is the number of points within the cloud, the complexity of voxelGrid downsampling is $O(k \log n)$ where k is the number of neighbors and n the number of points in the cloud. It is reasonable to assume that the averaging operations take $O(1)$ time since they can be done while searching. Statistical outlier removal searches for k nearest neighbors and removes those whose deviation is passed a certain threshold. Given that search operations take $O(\log n)$, for k neighbors, the complexity is $O(k \log n)$. The same assumption regarding the averaging computations is done here. A high amount of research has been done regarding the optimal complexity of RANSAC [51]. RANSAC has a complexity of $O(k + m_s * N)$ where k is the maximum amount of iterations defined by the user, m_s the average number of models per sample N the number of data points. FPFH operations have a complexity of $O(nk)$ as given in [22]. Finally, for the operation of color estimation, the k nearest neighbors are chosen and some constant operation is performed on them. The complexity here is similar to Statistical outlier removal since operations after the search are assumed to take $O(1)$ time. The complexity for color estimation then becomes $O(k \log n)$ where k is the number of neighbors, n the number of points.

The downsampling pipeline has a total complexity of:

$$O(\text{filter}) = O(k_0 \log n_{\text{init}} + k_1 \log n_1 + k_2 + m_s * n_2 + n_3 k_3 + k_4 \log n_3) \quad (9)$$

Different k indexes represent the number of neighbors defined for every operation. The n represents the number of points used as input. Using the notation of equation 9, n_{init} defines the whole cloud, n_1 the cloud after operation 1, n_2 the cloud after operations 2 etc.

Clustering: The complexity of the SMC sampler is defined in [33] as $O(TLKS N)$ where T defines the time frames, L the number of particles, K the number of clusters, S the number of samples, and N the size of the dataset. Given that the sampler is only used in single timesteps, the computational complexity is reduced to $O(LKS N)$. This is a low complexity and directly affects the performance since the sampler averages 10msec of execution time.

$$O(cluster) = O(LKSN)$$

Decision making: The decision making takes $O(\kappa * l^2)$ computational time where κ defines the number of clusters output by the sampler and l the number of landmarks currently stored in the database. This number can be further reduced by taking for example only landmarks that are nearby the cluster, but optimizing the decision making performance is outside the scope of this project.

Finally, with some notation abuse, the final complexity of the method can then be defined as:

$$\begin{aligned} O(filter) + O(cluster) + O(decision) = \\ O(k_0 \log n_0 + k_1 \log n_1 + O(k_2(t_M) + m_s * n_2) + n_3 k_3 + k_4 \log n_3) + O(LKSN_3) + O(\kappa * l^2) = \quad (10) \\ O(k_0 \log n_0 + k_1 \log n_1 + k_2(t_M) + m_s * n_2 + n_3 k_3 + k_4 \log n_3 + LKSN_3 + \kappa * l^2) \end{aligned}$$

The complexity as defined in equation 10 depends greatly on the initial reduction of the voxel downsampling. As the voxel leaf size parameter decreases and the downsampling outputs a larger cloud, the precision as well as the computing time of the method increases. Since in this thesis the research was directed towards online SLAM methods, the leaf size was modified so that the cloud the time requirements for online landmark tracking were met.

4.6. Landmark size

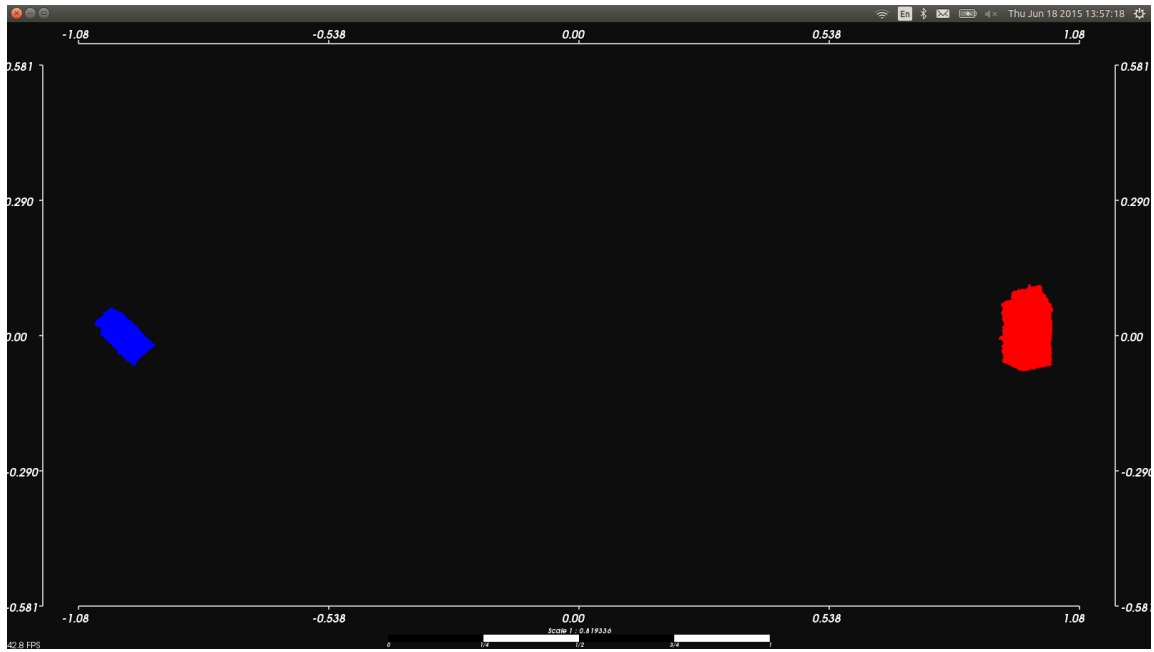
The basic block of this algorithm is the cluster. In order to be able to compute how scalable the method is, the size of a single landmark will be computed. Each landmark is represented by a single row in an SQLite database. To calculate how much space a landmark occupies, we will first calculate how space each of the columns has. Each row consists of 32 real numbers and 1 auto increment integer. Given that SQLite databases assign the memory dynamically, the worst case scenarios will be taken into account so that every other case is also included. According to the SQLite manual of datatypes [53] a real value takes up to 8 bytes of memory. An auto increment integer takes up to 4 bytes of information, so the total number of memory for a worst case scenario landmark is 260 bytes. This number can vary greatly over different environments and is a safe assumption to calculate the memory this method occupies.

5. RESULTS

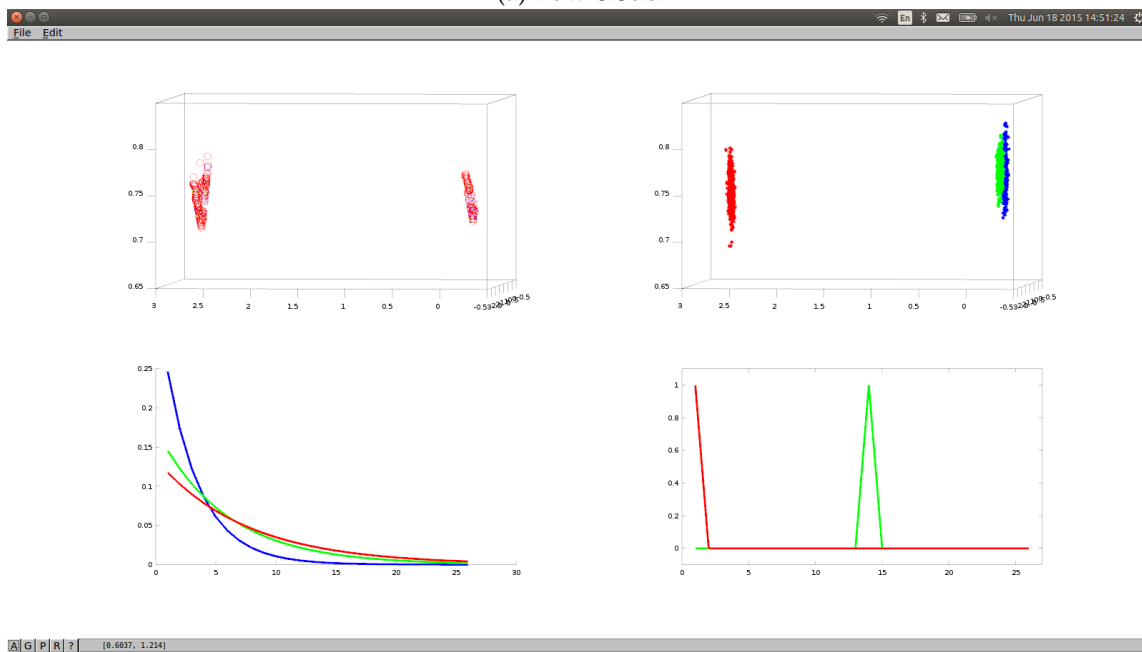
5.1. Simple datasets

In this section the algorithm will be tested against a simple dataset. That will make the conceptual extension to more complex– clouds that will be used when mapping an environment. In order to test the sampling as well as the decision layer of the algorithm, a simple dataset was used. The reduced cloud as well as the cluster assignments are color coded in the points and are shown in the top left Fig.5(b).

The cloud represents a milk cartridge transformed in two different positions. The pipeline will take the raw cloud as input, filter the cloud and pass it to the sampler to perform the clustering.



(a) Raw cloud



(b) Post clustering

Figure 5: Initial data along with the distributions inferred

In Fig.5(b). the reduced point cloud is shown in the top left part. It is significantly smaller in size and this cloud along with all the meta-information needed are used to perform the clustering. The sampler outputs a mixture of distributions that best fits the input data. The output of the cluster is shown in Fig.5(b) with the top right being the Gaussian distributions inferred, bottom left the exponential and bottom right the categorical representing the colour information of the cloud. The height of the objects leads to distribution with high variance in the z axis. The sampler outputs 3 clusters for the data and it can be seen that the change in angular information of the box leads the sampler to assign two distributions in the left milk box cloud. The color information is captured correctly in the structures and that was expected since the data had distinct colour signatures.

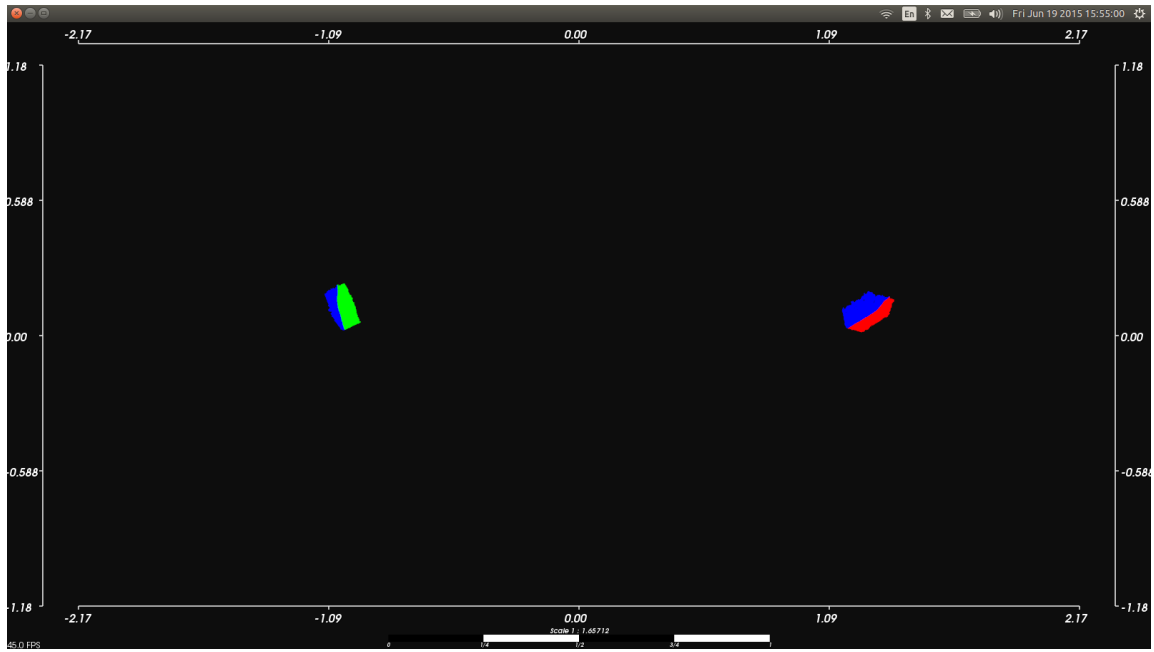
UUID 1					
LandUUID	GausKL	GausWes	ExpHel	ExpHel	CatKL
1	0	1.18111e-07	0	0.164045	0
2	0.0115821	53.5171	1.20726	0.319589	0
3	13.5579	22449.9	1.56956	0.376699	13.8155
UUID 2					
1	0.0115821	20.1454	1.7215	0.319589	0
2	5.42101e-20	4.88951e-08	0	0.411668	0
3	12.8379	21458.6	0.474719	0.449769	13.8155
UUID 3					
1	13.5579	31191.5	2.55205	0.376699	13.8155
2	12.8379	65013.8	0.53855	0.449769	13.8155
3	0	2.43346e-07	0	0.481313	0

Now that the structured is inferred, the distances between the clusters can be calculated. The table shows the distances between every cluster. It can be seen that the first two landmarks have small distances in their Gaussian counterparts since they represent nearby areas and objects in the cloud that are very far apart. The angular distances between them are also not very large. The distinct colors of the milk boxes make the color signatures of the clusters distinct. An example with more complex color signatures is shown in Fig.[6]. The clouds are now given a mixture of colors and this is passed to the color signature inferred on every cluster in Fig.[6](b)

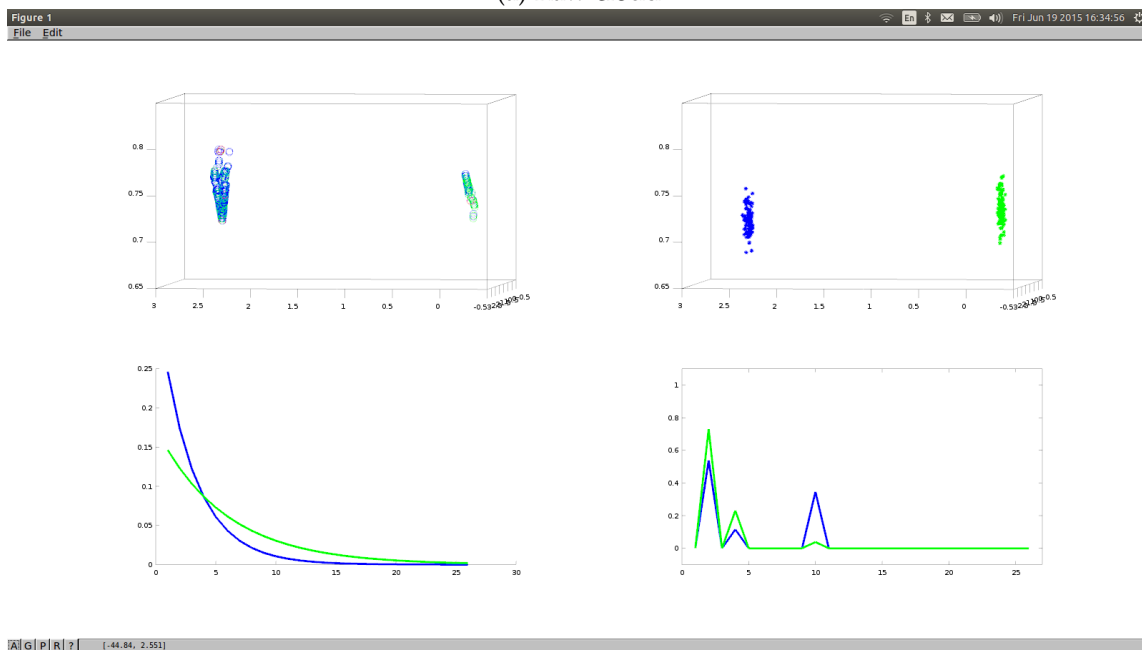
5.2. The decision layer

The decision boundary of the pipeline can be decomposed into three basic parts; a positional, an color and an angular boundary. Figure[7] shows the behavior of the decision layer with respect to the Gaussian(positional) and Categorical(color) parts.

In Fig.7(a) the yellow folder along with some of its environment to the left are being assigned to landmark 7386. The folder is then removed and a blue trashcan is put in its place. The folder and the trashcan are similar in size and due to that their Gaussian counterparts will not have large distances. Their main difference lies in the color signature they carry. Since the divergence in their color is substantial, a new landmark with id 7412 is created to capture the change in the signature of the environment at that place of the map. The different landmark assigned to the cluster can be seen in Fig.7(b). In Fig.7(c) and(d) the positional decision boundary is displayed.



(a) Raw cloud



(b) Post clustering

Figure 6: More complicated color distributions

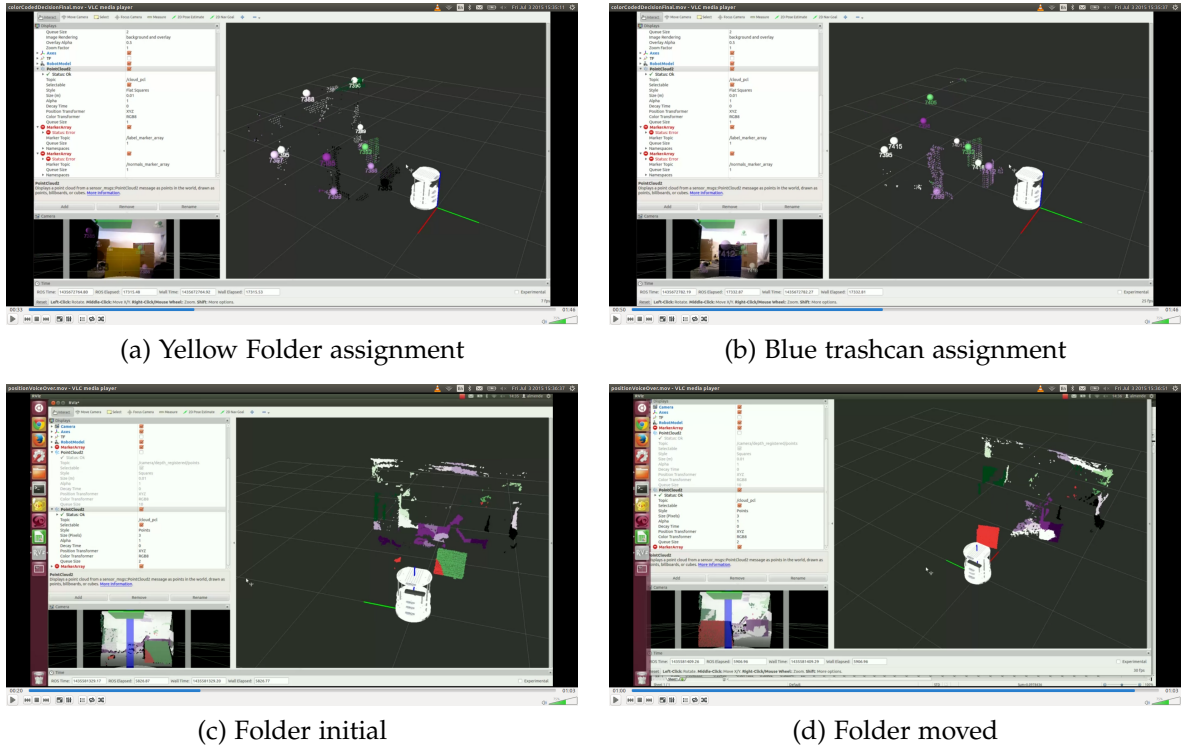


Figure 7: The colour and position boudnary is displayed in these pictures

In the initial position as shown in Fig.7(c) the yellow folder is assigned to the green landmark of the cloud. As the object is moved from position a to position b, it is being assigned to a different cluster. The reason the cluster is partly assigned to the red part of the cloud is due to the fact that the procedure is unsupervised. Due to that the folder is decomposed to several clusters and each one of them is being assigned to a different part of the cloud with respect to their position. This can be seen in Fig.7(c) where the bottom left of the folder is being assigned to the red cluster. Parameter tweaking is possible but the purpose of a general pipeline is to use as little parameter tweaking as possible so that the method is general and can be applied to multiple scenarios.

The exponential part of the distribution is responsible for the angle signature elements within a cluster have. In a more practical approach having a large amount of different angle distributions in a single cluster leads to objects that have texture and their surface is not smooth. Having a very strict limit in the angle distribution can lead to very small clusters and subsequently to a high amount of landmarks within the data. Using the data from the kinect camera, angle distributions are highly prone to high amounts of noise that is found in parts of the cloud that are near the end of Kinect sensor's range. Practical evaluation has shown that using an angle limit that is close to the average distance between angle signatures produces stable results and reasonably sized landmarks in the cloud.

5.3. EKF-SLAM experiments

To test the experiments in a real life scenario, the method was used as a sensor model in a Landmark based EKF SLAM algorithm. The method will be tested in its speed and memory requirements. As was shown when calculating the complexity of the method, speed complexity was shown to be very dependent to the downsampling process. If the downsampling threshold operations are strict and the cloud is being stripped of many of its points, accuracy will decline and speed will increase. As shown in the landmark size section, the unit cost of a Landmark has an upper limit. Therefore, the memory requirements of the method are directly depending on how many landmarks the algorithm allows before converging. A strict landmark matching decision layer that requires a high degree of correlation to match landmarks will converge to a higher amount of landmarks compared to a less strict one.

5.3.1 Speed

The speed factor was tested on a mid level as well as on a high level machine to see the differences in time needed to perform the operations. The results of a machine running on a pentium i5-3210M with 6GB of RAM are shown in table

Pentium i5-3210M					
Leaf Size	Downs Time	Cloud size	Sampling	Matching,old	Matching,new
1m	0.908768	9	0.0290296	0.00218081	0.00218081
50cm	0.914169	32	0.0112929	0.00154852	0.00243211
30cm	1.07862	79	0.0138666	0.00126673	0.00126673
10cm	1.1764	654	0.0712596	0.00154977	0.00154977
5cm	1.75907	2391	0.215594	0.00181846	0.00181846
1cm	14.3283	25556	5.06356	0.00278712	0.854803

The leaf size directly affects the time the cloud takes to be processed in the pipeline as well as the time the sampling takes afterwards. Experimental evaluation has showed that a leaf size between 1-5cm gives a good balance between speed and precision. The difference between Matching,old and Matching,new has to do with the fact that in the second case, a new landmark is added and therefore DB operations take place. When matching old is the same as matching new, then all the clusters were matched to items in the database.

Pentium i7					
Leaf Size	Downs Time	Cloud size	Sampling	Matching,old	Matching,new
1m	0.675993	7	-	-	-
50cm	0.779975	30	-	-	-
30cm	0.676315	66	-	-	-
10cm	0.746999	488	-	-	-
5cm	0.878036	1827	0.132831	0.00281951	0.00281951
1cm	1.11554	18487	2.69855	0.0021291	0.0021291

5.3.2 Memory requirements

As it is shown in the concentrated results that displays the features in combination with the point cloud, there is a large reduction in the dimensionality of the input. Given that a normal scene can be reproduced by an average of 100 clusters, and each cluster has a size of 280 bytes, the method is extremely scalable. Furthermore, the implementation can be extended to be even more performance based with techniques like recovering only the nearby clusters when searching for matches, sorting the database or using multi threading to increase performance. Utilizing such tools can make the technique being able to handle a very large amount of features since 100.000 clusters can be used to represent large areas take only 260MB of space.

6. CONCLUSION AND FUTURE WORK

A novel method for SLAM by using compressed representations of objects in a point cloud is introduced. Its strengths and weaknesses are presented. Future work could include:

- Improved tracking by using IMRF
- Extend to dynamic environments
- Extend the representation used

7. DISCUSSION

A fast method of pointcloud clustering and representation are presented. Its strengths and weaknesses are given. This method serves as a proof of concept on how non-parametric bayesians could help

REFERENCES

- [1] Thrun, S. (2002). Probabilistic robotics. *Communications of the ACM*, 45(3), 52-57.
- [2] Bailey, T., Nieto, J., Guivant, J., Stevens, M., & Nebot, E. (2006, October). Consistency of the EKF-SLAM algorithm. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on* (pp. 3562-3568). IEEE.
- [3] Thrun, S., & Montemerlo, M. (2006). The graph SLAM algorithm with applications to large-scale mapping of urban structures. *The International Journal of Robotics Research*, 25(5-6), 403-429.
- [4] Figueredo, A.J. and Wolf, P. S.A. (2009). Assortative pairing and life history strategy - a cross-cultural study. *Human Nature*, 20:317-330.
- [5] Thrun, S., & Mitchell, T. M. (1995). Lifelong robot learning. *The Biology and Technology of Intelligent Autonomous Agents*, 165-196.
- [6] Konolige, K., & Bowman, J. (2009, October). Towards lifelong visual maps. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on* (pp. 1156-1163). IEEE.

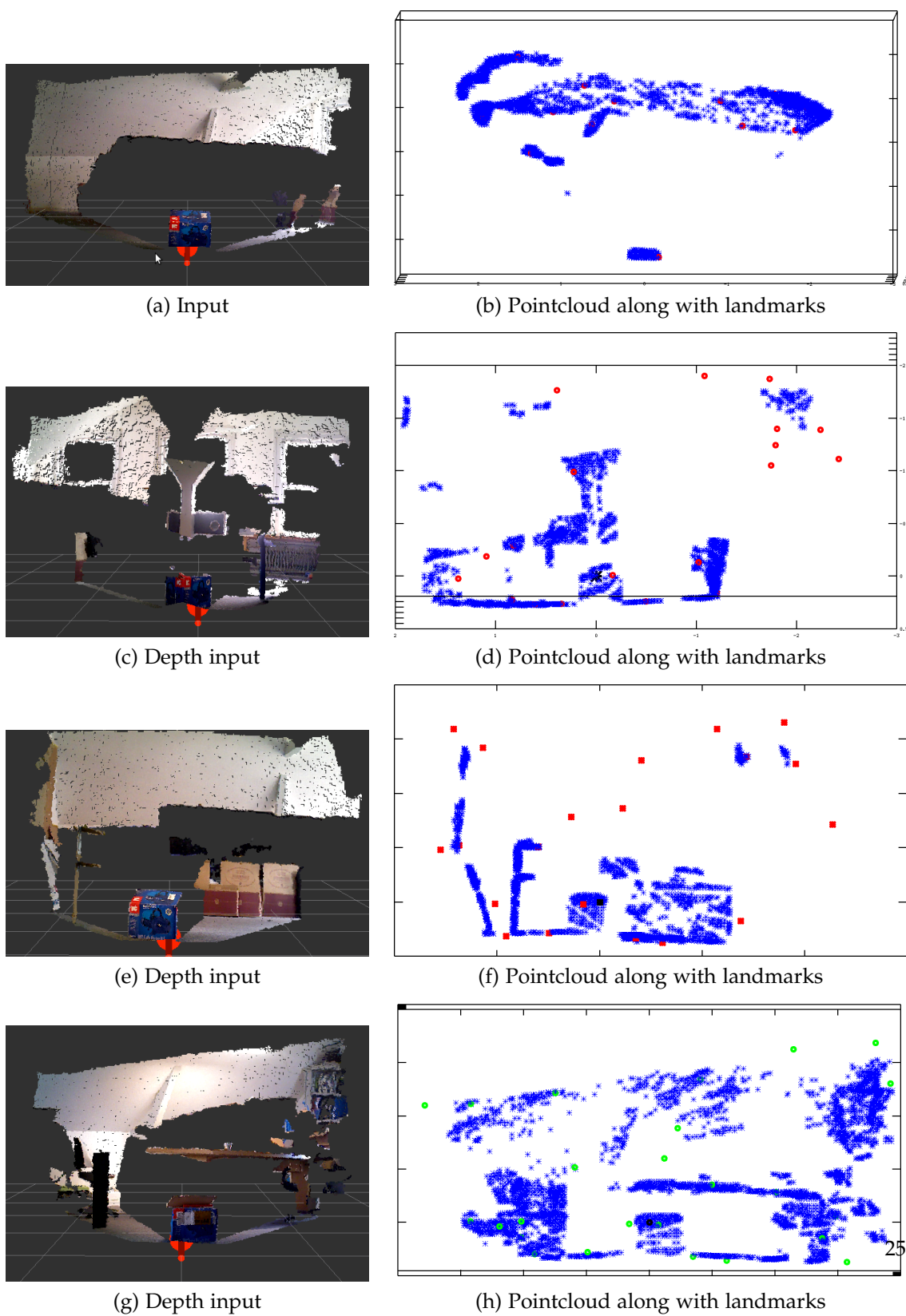


Figure 8: caption

- [7] Walcott, A. (2011). Long-term robot mapping in dynamic environments (Doctoral dissertation, Massachusetts Institute of Technology).
- [8] Hjort, N. L., Holmes, C., Mäijller, P., & Walker, S. G. (Eds.). (2010). Bayesian nonparametrics (Vol. 28). Cambridge University Press.
- [9] MacEachern, S. N. (2000) Dependent dirichlet processes. Unpublished manuscript, Department of Statistics, The Ohio State University.
- [10] Barber, D. (2012) Bayesian reasoning and machine learning.
- [11] Neiswanger, W., Wood, F., & Xing, E. The dependent dirichlet process mixture of objects for detection-free tracking and object modeling. In Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics (pp. 660-668) (2014, August)
- [12] Rusu, R. B., & Cousins, S. (2011, May). 3d is here: Point cloud library (pcl). In Robotics and Automation (ICRA), 2011 IEEE International Conference on (pp. 1-4). IEEE.
- [13] LabbÄl, M., & Michaud, F. (2011, September). Memory management for real-time appearance-based loop closure detection. In Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on (pp. 1271-1276). IEEE.
- [14] Salas-Moreno, R. F., Newcombe, R. A., Strasdat, H., Kelly, P. H., & Davison, A. J. (2013, June). Slam++: Simultaneous localisation and mapping at the level of objects. In Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on (pp. 1352-1359). IEEE.
- [15] Selvatici, A. H., & Costa, A. H. (2008). Object-based visual slam: How object identity informs geometry.
- [16] Castle, R. O., Gawley, D. J., Klein, G., & Murray, D. W. (2007, April). Towards simultaneous recognition, localization and mapping for hand-held and wearable cameras. In Robotics and Automation, 2007 IEEE International Conference on (pp. 4102-4107). IEEE.
- [17] Choudhary, S., Trevor, A. J., Christensen, H. I., & Dellaert, F. (2014, September). SLAM with object discovery, modeling and mapping. In Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on (pp. 1018-1025). IEEE.
- [18] Jensfelt, P., Ekvall, S., Kragic, D., & Aarno, D. (2006, September). Augmenting slam with object detection in a service robot framework. In Robot and Human Interactive Communication, 2006. ROMAN 2006. The 15th IEEE International Symposium on (pp. 741-746). IEEE.
- [19] Davison, A. J., Reid, I. D., Molton, N. D., & Stasse, O. (2007). MonoSLAM: Real-time single camera SLAM. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 29(6), 1052-1067.
- [20] Koo, S., Lee, D., & Kwon, D. S. (2014, September). Unsupervised object individuation from RGB-D image sequences. In Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on (pp. 4450-4457). IEEE.

- [21] Cichocki, A., & Amari, S. I. Families of alpha-beta-and gamma-divergences: Flexible and robust measures of similarities. *Entropy*, 12(6), 1532-1568.
- [22] Fast point feature histogram. Rusu, R. B., Blodow, N., & Beetz, M. (2009, May). Fast point feature histograms (FPFH) for 3D registration. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on* (pp. 3212-3217). IEEE.
- [23] Rabbani, T., van den Heuvel, F., & Vosselmann, G. (2006). Segmentation of point clouds using smoothness constraint. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 36(5), 248-253.
- [24] Caron, F., Davy, M., & Doucet, A. (2012) Generalized Polya urn for time-varying Dirichlet process mixtures. *arXiv preprint arXiv:1206.5254*.
- [25] Zhang, Z. (2012) Microsoft kinect sensor and its effect. *MultiMedia, IEEE*, 19(2), 4-10.
- [26] Wainwright, M. J., & Jordan, M. I. (2008). Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(1-2), 1-305. "
- [27] A.Trevor, J.Rogers, and H.Christensen. Omnimappper: A modular multimodal mapping framework. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2014
- [28] Koo, S., Lee, D., & Kwon, D. S. (2013, November). Multiple object tracking using an rgb-d camera by hierarchical spatiotemporal data association. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on* (pp. 1113-1118). IEEE.
- [29] Trevor, A. J., Gedikli, S., Rusu, R. B., & Christensen, H. I. (2013). Efficient organized point cloud segmentation with connected components. *Semantic Perception Mapping and Exploration (SPME)*.
- [30] Unnikrishnan, R., & Hebert, M. (2003, October). Robust extraction of multiple structures from non-uniformly sampled data. In *Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on* (Vol. 2, pp. 1322-1329). IEEE.
- [31] Rabbani, T., van den Heuvel, F., & Vosselmann, G. (2006). Segmentation of point clouds using smoothness constraint. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 36(5), 248-253.
- [32] Triebel, R., Shin, J., & Siegwart, R. (2010, June). Segmentation and unsupervised part-based discovery of repetitive objects. In *Robotics: Science and Systems* (Vol. 2).
- [33] Neiswanger, W., Wood, F., & Xing, E. (2014, August). The dependent dirichlet process mixture of objects for detection-free tracking and object modeling. In *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics* (pp. 660-668).
- [34] Cree, M. J., Jefferies, M. E., & Baker, J. T. Using 3D Visual Landmarks to Solve the Correspondence Problem in Simultaneous Localisation and Mapping.
- [35] Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2), 91-110.

- [36] Lamon, P., Tapus, A., Glauser, E., Tomatis, N., & Siegwart, R. (2003, October). Environmental modeling with fingerprint sequences for topological global localization. In *Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on* (Vol. 4, pp. 3781-3786). IEEE.
- [37] Sehgal, A., Cernea, D., & Makaveeva, M. (2010). Real-time scale invariant 3D range point cloud registration. In *Image Analysis and Recognition* (pp. 220-229). Springer Berlin Heidelberg.
- [38] Neal, R. M. (2000). Markov chain sampling methods for Dirichlet process mixture models. *Journal of computational and graphical statistics*, 9(2), 249-265.
- [39] Blei, D. M., & Jordan, M. I. (2006). Variational inference for Dirichlet process mixtures. *Bayesian analysis*, 1(1), 121-143.
- [40] Montemerlo, M., Thrun, S., Koller, D., & Wegbreit, B. (2002). FastSLAM: A factored solution to the simultaneous localization and mapping problem. *AAAI/IAAI*, 593-598.
- [41] Teh, Y. W., Jordan, M. I., Beal, M. J., & Blei, D. M. (2006). Hierarchical dirichlet processes. *Journal of the american statistical association*, 101(476).
- [42] Doucet, A., De Freitas, N., & Gordon, N. (2001). *An introduction to sequential Monte Carlo methods* (pp. 3-14). Springer New York.
- [43] Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent dirichlet allocation. *the Journal of machine Learning research*, 3, 993-1022.
- [44] MacEachern, S. N. (2000). -
- [45] Fox, E. B., Sudderth, E. B., Jordan, M. I., & Willsky, A. S. (2011). A sticky HDP-HMM with application to speaker diarization. *The Annals of Applied Statistics*, 5(2A), 1020-1056.
- [46] Charles E Antoniak, Mixtures of dirichlet processes with applications to bayesian nonparametric problems, *The annals of statistics* (1974), 1152-1174
- [47] F. Caron, M. Davy, and A. Doucet, Generalized Polya urn for time-varying Dirichlet process mixtures, *23rd Conference on Uncertainty in Artificial Intelligence (UAI-2007)*, Vancouver, Canada, July 2007, 2007
- [48] Fink, D. (1997). *A compendium of conjugate priors*.
- [49] ÅIjker, Y., GÄijnsel, B., & Cemgil, A. T. (2010). Sequential Monte Carlo samplers for Dirichlet process mixtures. In *International Conference on Artificial Intelligence and Statistics* (pp. 876-883).
- [50] Del Moral, P., Doucet, A., & Jasra, A. (2006). Sequential monte carlo samplers. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(3), 411-436.
- [51] Meer, P., Mintz, D., Rosenfeld, A., & Kim, D. Y. (1991). Robust regression methods for computer vision: A review. *International journal of computer vision*, 6(1), 59-70.

[52] Raiffa, H. (1974). Applied statistical decision theory.

[53] <https://www.sqlite.org/datatype3.html>