Towards Bayesian lifelong learning in mapping

Panagiotis Chatzichristodoulou, Rico Möockel, Kurt Driessens, Anne Van Rossum

University of Maastricht, DoBots B.V.

1 Introduction

Simultaneous localization and mapping is one of the fundamental problems of autonomous systems[1]. In order for robots to be truly autonomous they have to be able to enter an environment and map 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] and FastSlam[26] we can now efficiently map such environments. The logical extension to methods that can map static environments is methods that remove this restriction. The idea of lifelong robot learning was introduced as a general concept to the literature by Sebastian Thrun[3]. Konolige et al.[4] specifically focus on lifelong learning in mapping. In the PhD thesis of Walcott[5] long term mapping methods are decomposed to 4 basic subproblems: the problem of being able to continuously incorporate new information, tracking the growing DPG, detecting changes and update the map online, being able to handle changes to the map as changes occur with the passage of time.

The first two problems can be though of as compression problems as the map increases over time whereas the latter ones can be though of as dynamic environment problems. In this project the focus will be directed towards slam methods that use RGBD devices like Microsoft's Kinect to perform slam. 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 the solution.

Dirichlet processes and Dirichlet process mixture models[13] 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 main motivation of this thesis is to integrate such methods to mapping algorithms as a means of creating compressed representations of the environment that also retain enough expressivenes so that they can be used as reference points when navigating through the environment. That would tackle the compression subproblem of long term mapping and would be a solid first step towards a general Bayesian solution to the long term mapping problem.

The rest of the paper is structured as follows. In Section 2 relevant literature review will be reviewed, Section 3 will introduce the basic background theories of the model, Section 4 will define the method presented in this paper, Section 5 will show experimental results of the method. Finally, conclusion and future directions are presented in Section 6.

2 Literature review

Literature review will be focused on 4 related sub fields: Object based slam or semantic slam, point cloud object segmentation, non-parametric clustering methods ad the correspondence problem in slam.

2.1 Object based slam

Salas-Moreno et al.[7] define a method of performing object based slam for specific classes of 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 where the robot knows what objects to expect. Castle et al. use object recognition to perform object based slam with the use of a hand-held cameras. Selvatici et al.[8] 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.[9] use point clouds and an object database to match objects currently seen with known objects within their database. They use omnimaper[14] 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. Seongyong Koo et al.[10] introduce a method of unsupervised object individuation from RGB-D image sequences. They cluster their initial cloud into candidate objects using Euclidean clustering and proceed to extract features like the Euclidean distance(L2) and the

Kullback-Leibler distance between point cloud objects. They use IMFT to solve their tracking problem.

2.2 Point Cloud Object clustering

Trevor et al.[15] take positional information, Euclidean distances and the normal of points to as input to their functions and output segments that are part of the same object. PCL library[6] 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 [16],[17] using cues like normals and curvature. Rabbani et al.[12] 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.[18] 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 non-parametric statistics. Radford M. Neal[24] 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. Variational inference for DPMM's, introduced by Jordan et al.[25] 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. Particle filter approaches of inference have also been established with Doucet et al.[27] introduce Sequential Monte Carlo as a fast way to approximate inference. For the purpose of this paper an SMC sampler will be defined in detail in the Model definition section.

2.4 Correspondence

Under the semantic slam context, correspondence refers to the problem of identifying objects as ones that have been encountered before during the mapping process. Towards that direction Cree et al.[20] 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.[21] match Scale Invariant Feature Transform (SIFT) features, an approach which transforms image data into scale-invariant coordinates relative to local features. Lamon et al.[22] store a database of fingerprints which indicate the location in the robot's environment. The features are ordered and stored at a database at 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.[23] an extension of SIFT descriptors to 3D data and point clouds is given.

3 Theory background

The basic theory background regarding the sampler will be presented in this section. Generalized Polya Urn is an extension on basic Urn[29] models for Dirichlet processes and serves as the base of the sampler presented in the theory section.

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[28]. Dependent dirichlet processes extend those tools by allowing the clusters in the data to vary with some variance over time by introducing dependencies on the data which can be temporal, positional etc. The DDPs are a natural extension of the DP's in domains where data cannot be considered exchangeable. The main motivation behind using such tools is that they can naturally be extended to dynamic environments to tackle the dynamic part of the long term slam problem.

A DDP also known as Generalized Polya Urn(GPU)[29] and has the property of randomly deleting partitions of clusters on every iteration. That way, it can cope with the variance of the data. The current notation defines the n_{th} datapoint at time t,$x_{t,n}$ having an assignment $c_{t,n}$ at cluster $k \in \{1, 2, ..., K\}$. 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** $k = 1, ...K_{t-1,N_{t-1}}$ **do**
3: Draw $\Delta s_{t-1}^k \sim Binom(s_{t-1,N_{t-1}}^k, \rho)$
4: Set $s_{t,0}^k = s_{t-1,N_{t-1}}^k - \Delta s_{t-1}^k$
5: **end for**
6: **for** $n = 1, ...N_t$ **do**
7: Draw $c_{t,n} \sim Cat(\frac{s_{t,n-1}^1}{\alpha+\sum_k s_{t,n-1}^k}, \frac{s_{t,n-1}^{K_{t,n-1}}}{\alpha+\sum_k s_{t,n-1}^k}, \frac{\alpha}{\alpha+\sum_k s_{t,n-1}^k})$
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 though of in the terms of a general chinese restaurant process[28] as shown in Fig. 1. At time t, suppose there are n 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).

4 Model definition

4.1 General pipeline

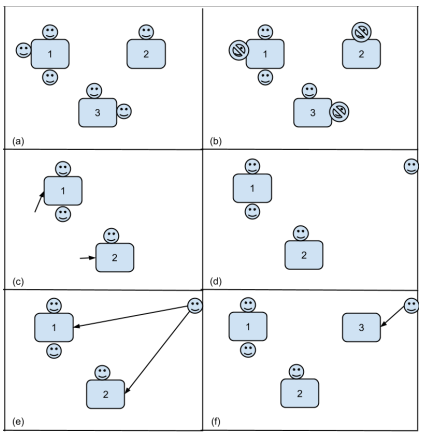


Figure 1: General Polya

The general flow of operations that occur in the EKF module is presented in Fig. 2. The slam module requests new observation readings given the cloud currently read by the sensors and the position of the robot. The pipeline takes that cloud, extracts clusters and returns the landmarks currently being observed while taking into account landmarks that where observed in the past. Landmarks and clusters are identical concepts representing a different layer in the pipeline. More specifically, clusters are output from the sampler from the sampler and are given as an input of landmarks to the EKF module.

Method input: The method takes as input a point cloud as it is currently being read by the kinect sensor.

Lines 3-4: The preprocessing the cloud is done in these steps. Feature extraction is done through the pcl[6] library. A voxel grid is used to reduce the dataset size. A leaf size of approximately 3cm produces a good trade-off between precision and speed. The object representation used approach is similar to[9]. Instead of using the CSHOT descriptor, fpfh[11] histogram is used instead. A fast point feature histogram(fpfh) represents an angular signature between a point and its neighbors. The color signature of the cloud is being encoded with an approach similar to [19]. The color spectrum is discretized and what is extracted is the count of different color signatures between a point and its k nearest neighbors. Finally the position of every point is also given as input to the algorithm. The pipeline is presented in figure Fig. 2. The preprocessing outputs a vector of $\mathbf{x} = (x_s, x_c, x_a)$ where s represents the

Algorithm 2 Landmark Layer

1: **procedure** GETLANDMARKIDS($pointCloud, timepoint, existingLandmarks$)
2: $initialize(landMarkIds)$
3: $pointCloudReduced \leftarrow extractMetaFeatures(pointCloud)$
4: $features \leftarrow extractMetaFeatures(pointCloudReduced)$
5: $landmarks \leftarrow cluster(features)$
6: **for** $landmarks$ `as` $landmark$ **do**
7: $(similarity, landId) \leftarrow calcBestSim(landmark, existingLandmarks)$
8: **if** $similarity > threshold$ **then**
9: $addLandmarks(landMarkIds, landId)$
10: **else**
11: $newLandID \leftarrow addLandmarkDB(landmarkDB, landmark)$
12: $addLandmarks(newLandID)$
13: **end if**
14: **end for**
15: **return** $landMarkIds$ ▷ Return landmarks
16: **end procedure**

position information of the point, c the color information of the point's neighbors and a the angle information of the point's neighbors.

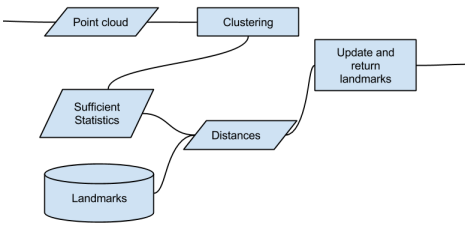


Figure 2: Pipeline flow

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. The clustering is done using the SMC sampler presented in the model definition section.

Lines 6-12: The correspondence of previously seen landmarks to current observations is computed in lines 6-12. The calcBestSim function returns the landamrk with the highest similarity match with the landmarks already stored in the database.

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

Now that the basic theories are defined, the pipeline's components and its operating mechanisms can be presented in more detail.

4.2 The sampler

4.2.1 The data distribution

Each point x in the cloud is represented as a tuple $x = (x^s, x^a, x^c)$ where superscript s represents spatial information, a angle information, and c color information. The method those features are extracted is explained in lines 3 and 4 in the general pipeline section. For the purpose of this project each point in the cloud is represented by vector of length 33 with the first three elements representing the space information, elements 4-6 angle information, and the rest color information.

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

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

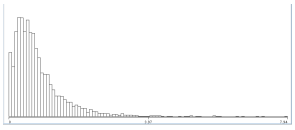


Figure 3: Exponential trend of distances

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 the weights of the color distribution and Exp is an exponential 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 as shown in Fig. 3.

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:

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

Given the theory in [29], the transition Kernel must satistfy:

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

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[31] 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 $\theta_t^k \sim T(\theta_{t-1}^k) = T_2 \circ T_1(\theta_{t-1}^k)$ with:

$$\begin{aligned} z_t^k \sim T_1(\theta_{t-1}^k) &= Normal(\mu_{t-1}, \Sigma_{t-1})Mult(\delta_{t-1})Exp(\lambda_{t-1}) \\ \mu_t, \Sigma_t, \delta_t, \lambda_t \sim T_2(z_t^k) &= NiW(\kappa_0, \mu_0, \nu_0, \Lambda_0)Dir(q_0)Gam(\alpha_0, \beta_0) \end{aligned} \tag{1}$$

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

4.2.2 Sequential monte carlo sampler

Sequential monte carlo samplers for Dirichlet process mixture models where introduced by Doucet et al [32] and serve as fast alternative to MCMC and VI methods of performing posterior inference. Given the previous definitions, the sampler is now defined as:

4.2.3 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:

$$Q_1(c_{t,n}|s_{t,n}^k, \theta_t^k, \alpha) \propto Cat(s_{t,n}^1, ...s_{t,n}^K, \alpha) \times \begin{cases} F(x_{t,n}|\theta_t^{c_t}) & \text{if } k \leq K_{t-1} \\ \int P(x_{t,n}|\theta_t)G_0(\theta)d\theta & \text{if } k > K_{t-1}. \end{cases} \tag{2}$$

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:

$$\begin{aligned} \int P(x_{t,n}|\theta_t)G_0(\theta)d\theta = t_{\nu_0-1}(x_{t,n}^s|\mu_0, \frac{\Lambda_0(\kappa_0+1)}{\kappa_0(\nu_0-1)}) \times \prod_{j=1}^{V} \frac{\Gamma(x_{t,n}^c)}{\Gamma(q_0)} \times \\ \frac{\Gamma(\sum_{j=1}^V q_0)}{\Gamma(\sum_{j=1}^V x_{t,n}^c)} \times Lomax(\alpha_0 + s_{t,n}^c, \beta_0 \sum_{j=1}^{V} x_{t,n}^c) \end{aligned} \tag{3}$$

Algorithm 3 SMC for DDPM

Input: Points $\{x_{1,1:N_t}, ..x_{T,1:N_t}\}$with extracted features
Output: Clusters representing of the data
for $t = 1, ...T$ **do**
  for $l = 1, ...L$ **do**
    for $iter = 1, ...S$ **do**
      Sample $(c_t)^{(l)} \sim Q_1$
      Sample $(\theta^k) \sim Q_2$
    end for
  end for
  for $k = 1, ...K$ **do**
    Sample $\Delta s^k_{t-1} \sim Binom((s^k_{t-1,N_{t-1}})^{(l)}, \rho)$
    Set $s^k_{t,0} = s^k_{t-1,N_{t-1}} - \Delta s^k_{t-1}$
    Sample $((z^k_{t+1})^{(l)}) \sim T_1((\theta^k_t))^{(l)}$
  end for
  compute particle weights w^l_t
end for
Normalize and resample weights

Where t represents student's t-distribution with ν degrees of freedom, Lomax represents Lomax distribution with shape and scale, α and β repsectively and the rest represent a Dirichlet-Multinomial(aka DirMul) distribution. The formulas of the posterior predictive distributions can be found in the literature with [30] 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:

$$\begin{aligned} Q_2(\theta^k_t|\theta^k_{t-1}, x^k_t, z^k_t) &\propto F(x^k_t|\theta_k) \times T_2(\theta^k_t|z^k_t) \\ = NiW(\mu^k_t, \Sigma^k_t|\kappa_n, \mu_n, \nu_n, \Lambda_n)Dir(\delta^k_t|q_n)Gam(\lambda^k_t|\alpha_n, \beta_n) \end{aligned} \tag{4}$$

With:

$$\begin{aligned} \kappa_n = \kappa_0 + N, \nu_n = \nu_0 + N, \mu_n = \frac{\kappa_0}{\kappa_0 + N}\mu_0 + \frac{N}{\kappa_0 + N}\overline{x}^s \\ \Lambda_n = \Lambda_0 + s^s_x, q_N = q_0 + \sum_n x^c_i, \alpha_n = \alpha_0 + N, \beta_n = \beta_0 + \sum^n x^a_i \end{aligned} \tag{5}$$

Where $\overline{x}$ 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 cojugate priors like[30].

4.2.4 Weight updates

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

$$w^{(l)}_t = \frac{P(c^{(l)}_t, \theta^{(l)}_t, x_t|\theta_{t-1})}{P(c^{(l)}_t, \theta^{(l)}_t|\theta_{t-1})} \tag{6}$$

Using Bayes rule, the numerator can be written as:

$$P(x_t, |c^{(l)}_t, \theta^{(l)}_t\theta_{t-1}) \times P(c^{(l)}_t, \theta^{(l)}_t|\theta_{t-1}) \tag{7}$$

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.

4.3 Decision Layer

The decision layer calculates how similar a cluster is to ones encountered before. To achieve that, distance measures must be defined between the stored clusters and the ones that are inferred at the current iteration of the algorithm. Distances between distributions are called divergences and a large amount of literature on divergences exists.

Every cluster consists of a three part distribution as it was defined in section 4.2.1. Now let l be the distribution of a stored cluster and o the distribution of a currently observed cluster. l and o can be decomposed into 3 parts: l_G,l_C,l_E where G,C and E stand for Gaussian, Categorical and Exponential respectively. 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: Wesserstein(l_G, o_G), Kullback-Leibler(l_G, o_G), SquaredHellinger(l_E, o_E), Kullback-Leibler(l_E, o_E), Kullback-Leibler(l_C, o_C).

The distance between every distribution can be transposed of a vector of length 5. That way, deciding if a cluster is part of a landmark that has been encountered before is now a problem of finding the optimal decision boundary given the distances at hand. For this project, the decision boundary by empiric evaluation of the landmarks.

4.4 Complexity

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

$$O(total) = O(filter) + O(cluster) + O(decision)$$

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:

$$O(filter) = O(Downsampling + Stat\ Removal + RANSAC + FPFH + Color\ estimation)$$

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 $O(klogn)$ where k is the number of neighbors and n the number of points in the cloud. Statistial outlier removal searches for k nearest neigbhors and removes those whose deviation is passed a certain threshold. Searching for k neighbors in cloud has a complexity of $O(k\ log\ n)$.A high amount of research has been done regarding the optimal complexity of RANSAC [33]. 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 [11]. Finally, for the operation of color estimation, the k nearest neighbors are chosen and some constant operation is performed on them. 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:

$$O(filter) = O(k_0\ log\ n_{init} + k_1\ log\ n_1 + k_2 + m_s * n_2 + n_3 k_3 + k_4\ log\ n_3) \tag{8}$$

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 8, 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 [19] as $O(TLKSN)$ 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.

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.

The final complexity of the method can then be defined as:

$$O = O(k_0 logn_0 + k_1 logn_1 + k_2(t_M) + m_s * n_2 + n_3 k_3 + k_4 logn_3 + LKSn_3 + \kappa * l^2) \tag{9}$$

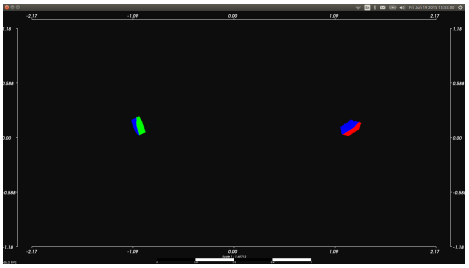
4.5 Landmark size

The basic block of this algorithm is a cluster containing an environment signature. 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. Each row consists of 32 real numbers and 1 auto increment integer. Given that SQLite databases assign the memory dynamically, the maximum amount of memory a cell can take will be calculated so that every other case is also included. According to the SQLite manual of datatypes 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.

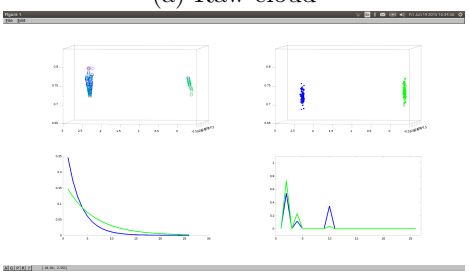
5 Results

5.1 Simple datasets

In this section the algorithm was tested against a simple dataset. Testing the method on simple datasets will make easier the extension to more complex clouds that will be used when mapping the environment. In order to test the sampling as well as the decision layer of the algorithm, a simple dataset provided by the pcl[6] library was used. More specifically, a cloud consisting of two milk cartridges with different colors and poses where used. The initial cloud is shown in Fig.5.1(a). The cloud was given as input to the downsampling pipeline. The reduced cloud was then passed as input to the sampler and the clustering results are shown in Fig.5.1(b).



(a) Raw cloud



(b) Post clustering

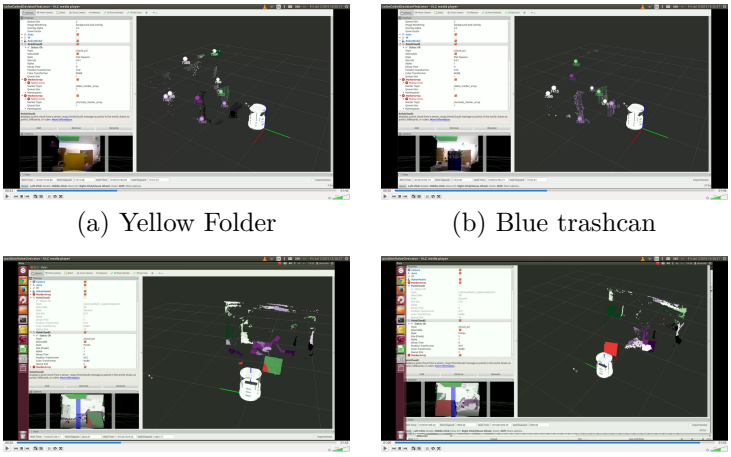
More specifically, the reduced point cloud is shown in the top left part. It is significantly smaller in size and this cloud along with all the meta-information needed to perform the clustering are given as input to the sampler. The sampler outputs a mixture of distributions that best fits the input data. The clustering output is shown in Fig.5.1(b) with the top right being the Gaussian distributions inferred, bottom left the exponential and bottom right the categorical representing the color information of the cloud. The height of the objects leads to distribution with high variance in the z axis. The sampler outputs 2 clusters for the data with each box being assigned separately. The color signature each cluster carries is correctly captured in the bottom right part of Fig.5.1(b). The two boxes are similar in size but their orientation is different and this slight difference is shown in the exponential part of the signature. Each box is now captured as an environment signature that consists of a Gaussian, a Categorical, and an Exponential part.

UUID 1					
LandUUID	GausKL	GausWes	ExpHel	ExpHel	CatKL
1	0	1.18111e-07	0	0.164045	0
2	13.5579	22449.9	1.56956	0.376699	13.8155

Table 5.1 shows the distances between every cluster computed against the first. The Gaussian counterparts of the clusters have significant distances due to the distance every cluster has in space. On the other hand, the Distances between their Exponential parts are not that large and the overlap between the categorical singatures leads to medium distances in their Categorical counterparts. Those distances are the information that is being passed to the decision layer to match existing landmarks with new ones.

5.2 Expresiveness and decision layer

Post clustering, of operations in the pipeline have to do the comparison of clusters currently extracted with landmarks already stored in the database. The accuracy of the operations is dependent on the expressiveness of the signatures the clusters carry and this is displayed in the examples presented in Fig.4. The decision boundary of the pipeline can be decomposed into three basic parts; a positional, a color and an angular boundary. Fig.4 shows the behavior of the decision layer with respect to the Gaussian(positional) and Categorical(color) parts.



(a) Yellow Folder (b) Blue trashcan

(c) Folder initial (d) Folder moved

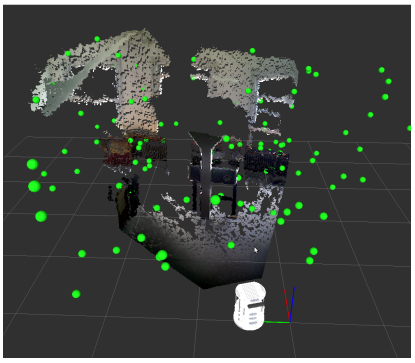
Figure 4: Expresiveness & decision bounds

In Fig.4(a) the yellow folder allong with some of its environment to the left are being assigned to landmark with id 7383. 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 distance 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.4(b). In Fig.4(c) and(d) the positional decision boundary is displayed. In the initial position as shown in Fig.4(c) the yellow folder is assigned to the green landmark of the cloud. As the object is moved a different position in the cloud, it is being assigned to a different cluster. The reason the cluster is assigned to multiple landmarks is due to the fact 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.4(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 easily be applied to a different scenarios.

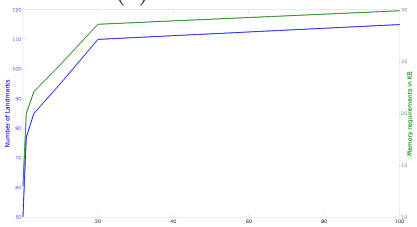
The exponential part of the distribution is responsible for the angle signature elements within a cluster have. Practically, having a large amount of different angle distributions in a single cluster leads to objects that have texture and their surface is not smooth. Having a very strict limit in the 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 sensitive to 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

Finally, the pipeline was used in real life scenarios as a sensor model in a Landmark based EKF slam algorithm and was tested in its precision and memory requirements. In Fig.5 an end result of a slam session using the pipeline as a sensor model is shown. As the robot progresses through the environment, the EKF module requests from the sensor model to observe what landmarks are currently detected given the current cloud readings and the existing Landmark database. The pipeline follows the procedure defined in the general pipeline section and returns the landmarks currently detected. The end results shows the amount of landmarks the method requires in order to map a medium sized room. Each sphere represents an environment signature at that specific point the same way it was shown in the simple datasets sections. If the decision layer is chosen to be very strict the number of landmarks the algorithm outputs will increase, and the environment will be represented in more detail. It is important to notice that a strict decision layer must be handled with care as it can lead to a pipeline that continuously adds new landmarks making the sensor module non-converging.



(a) Slam session



(b) Memory requirements

Figure 5: SLAM session and memory requirements

The compression that method introduces is not directly observable from the figure. As the environment is being reduced from a cloud to a landmarks, the memory needs change from using point clouds to using the landmarks extracted in those clouds. Since every landmark represents a signature of the environment at this particular point, the compression is done by reducing a high amount of points to that specific signature. The number of parameters needed to define the three distributions in the signature is all the information this method requires and hence the memory gains are substantial. The expresiveness of the distributions greatly affects the objects as well as object shapes that could be represented. Finally it is always possible to extend the method presented by adding new features and new priors to those features.

5.4 Memory requirements

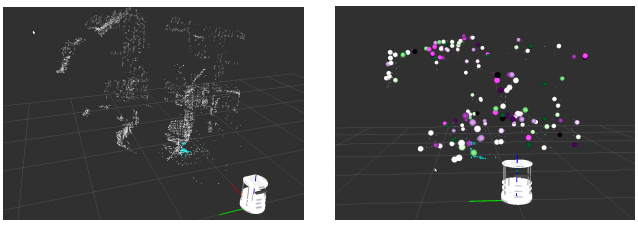
The memory requirements of the method is related to the Dirichlet strength parameter α of the sampler. As the α parameter increases, the sampler will output more clusters on every iteration. The higher amount the of cluster will result a higher amount of landmarks and, consequently, in larger memory requirements in the method. Fig.5(b) shows memory requirements as a function of strength parameter alpha.

As it can be seen, the number of landmarks follows the logarithmic trend of the Dirichlet prior in relation to α. Constantly increasing the alpha value will not make the algorithm follow the logarithmic trend indefinitely. That is due to the fact that despite the alpha increase, the decision layer has an upper bound to the number of landmarks it can have and will eventually saturate. It must also be noted that as the alpha parameter is set to higher numbers, the sampler outputs more clusters making it a more accurate environment descriptor, but also takes more time, making it non feasible to use in real time scenarios. Values of alpha between 0 and 10 provide a robust and fast enough sampler that can be used in online mapping scenarios.

5.5 Limits of the method

Limits of the method exist in relation to the two basic layers of the pipeline. The clustering layer and the decision making.

5.5.1 Clustering layer



(a) Low α (b) High α pipeline

Figure 6: Pipeline fail cases.

An important limit of the pipeline exists with respect to the Dirichlet hyper-parameter α. During the clustering, choosing a correct value for the hyper-parameter α is very important in order to have optimal results. Having sampler run with a very low α can lead to the whole cloud being assigned a single cluster. Having every point in the cloud being part of the same cluster leads to a a significant amount of information loss as no environment specific information are incorporated to the cloud. Fig.[6](a) shows the behaviour of the sampler for α values lower than 1.

On the other hand, having a very large α can lead to a very large amount of clusters being output by the sampler every time. That can lead to a non-converging pipeline since every time a new landmark that does not fit the landmark database is output. This leads to many small clusters each containing small number of points each. Furthermore, having a high value in the hyperparameter leads to a slower sampling procedure since the complexity of the sampler is $O(TKLSN)$ where K

is the number of clusters. Fig.[3](b) displays the behavior of the sampler for very large values α. The spheres represent landmarks and it can be seen that most of the points in the environment are considered individually clusters.

5.5.2 Decision layer

The restrictions imposed by the decision layer are straightforward due to the constant nature of the decision process. Taking very small distances in the distance threshold operation can lead to pipeline that continuously adds new landmarks to the database. An example is shown in Fig. 7 where limiting the landmark matching operation to very small exponential distances, leads to a cloud where objects are decomposed to a lot entitites.

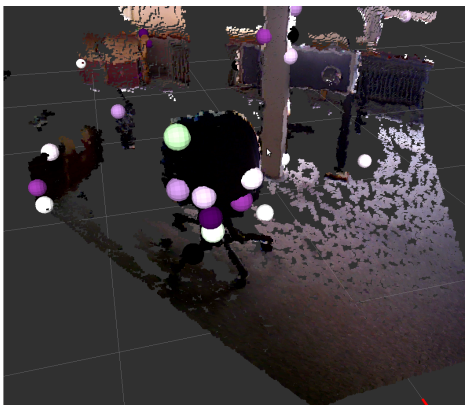


Figure 7: Strict exponential limit

Object decomposition is even more intense when camera movement is involved due to the different angles. Finally, since the number of landmarks is also a function of noise, areas of the cloud that are near the maximum range of the sensor can lead to different landmarks added frequently.

6 Conclusion and future work

In this paper a novel method for cloud representation by using non-parametric Bayesian tools was introduced. By reducing parts of the environment to distribution signatures, a large amount of compression is provided making the method highly suitable to target long term slam problems; the expresiveness of the representation is enough to perform slam and is robust even in the noisy parts of the environment. The strengths and weaknesses of the method were presented and an application on how this method could be used to tackle the compression problem of long term slam were given.

There is a number of directions which could be explored to improve the method. Choosing more complex environment representations could increase the expresive strength of the sampler making it easier to represent with higher precision more complex structures. Also, a hierarchical clustering approach would be an interesting extension since it would both have the ability to capture the structure in a top-down maner and as well as handle the dynamic component that long term slam problems introduce. Furthermore, having a more complex decision layer would also increase the robustness of the method making it able to handle more complex environment structures and more expresive landmarks. Finally, those additions could help lift the dynamic environment restrictions that were defined in Walcotts PhD thesis and make the pipeline tackle in a fully Bayesian way the implications of life-long slam.

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., & Mitchell, T. M. (1995). Lifelong robot learning. The Biology and Technology of Intelligent Autonomous Agents, 165-196.

[4] 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.

[5] Walcott, A. (2011). Long-term robot mapping in dynamic environments.

[6] 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.

[7] 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.

[8] Selvatici, A. H., & Costa, A. H. (2008). Object-based visual slam: How object identity informs geometry.
[9] 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.
[10] 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.
[11] 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.
[12] 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.
[13] 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. "
[14] A.Trevor, J.Rogers, and H.Christensen. Omnimapper: A modular multimodal mapping framework. In IEEE International Conference on Robotics and Automation (ICRA), 2014
[15] 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
[16] Unnikrishnan, R., & Hebert, M. (2003, October). Robust extraction of multiple structures from non-uniformly sampled data. In Intelligent Robots and Systems, 2003. Proceedings. 2004 IEEE/RSJ International Conference on IEEE.
[17] 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.
[18] Triebel, R., Shin, J., & Siegwart, R. (2010, June). Segmentation and unsupervised part-based discovery of repetitive objects. In Robotics: Science and Systems (Vol. 2).
[19] 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).
[20] Cree, M. J., Jefferies, M. E., & Baker, J. T. Using 3D Visual Landmarks to Solve the Correspondence Problem in Simultaneous Localisation and Mapping.
[21] Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. International journal of computer vision, 60(2), 91-110.
[22] 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. Proceedings. 2003 IEEE/RSJ International Conference on IEEE.
[23] 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.
[24] Neal, R. M. (2000). Markov chain sampling methods for Dirichlet process mixture models. Journal of computational and graphical statistics, 9(2), 249-265.
[25] Blei, D. M., & Jordan, M. I. (2006). Variational inference for Dirichlet process mixtures. Bayesian analysis, 1(1), 121-143.
[26] Montemerlo, M., Thrun, S., Koller, D., & Wegbreit, B. (2002). Fastslam: A factored solution to the simultaneous localization and mapping problem. AAAI/IAAI, 593-598.
[27] Doucet, A., De Freitas, N., & Gordon, N. (2001). An introduction to sequential Monte Carlo methods (pp. 3-14). Springer New York.
[28] Charles E Antoniak,Mixtures of dirichlet processes with applications to bayesian nonparametric problems, The annals of statistics (1974), 1152âĂŞ1174
[29] F. Caron, M. Davy, and A. Doucet, Generalized Polya urn for time-varying Dirichlet process mixtures, 23rd Conference on Uncertainty in Artificial Intelligence (UAIâĂŹ2007), Vancouver, Canada, July 2007, 2007
[30] Fink, D. (1997). A compendium of conjugate priors.
[31] ÃIJlker, Y., GÃijnsel, B., & Cemgil, A. T. (2010). Sequential Monte Carlo samplers for Dirichlet process mixtures. In International Conference on Artificial Intelligence and Statistics
[32] 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.
[33] 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.