Chapter 2

Related Work

Contents In robotics depth sensors generate point clouds. The tasks of robotic object recognition, positioning, and navigation require models that represent such point clouds. It is unclear whether the current methods that perform inference over point clouds are appropriate for these tasks. The current models do not model uncertainty explicitly. This chapter presents models that can be used for point cloud modeling and that represent uncertainty. This (partially) answers research question RQ1. The chapter concludes with recommendations for the development of point cloud inference models. They will be implemented in a new model for line inference in Chapter 3 and line segment inference in Chapter 4.

Outline This chapter describes one particular nonparametric Bayesian model, the Dirichlet process. The process is presented in four ways: (1) as a measure, (2) with a sequential representation (using a restaurant metaphor), (3) as a prior for a mixture, and (4) exhibiting a stick-breaking distribution for cluster sizes (Section 2.1). Six inference methods are described: (1) inverse transform sampling, (2) rejection sampling, (3) approximate Bayesian computation, (4) Gibbs sampling, (5) Metropolis-Hastings sampling, and (6) Split-Merge Markov chain Monte Carlo sampling (Section 2.2). Inference about point clouds in the chapters to follow will use adaptations of the described models and inference methods for which some recommendations are given (Sections 2.3 and 2.4) .

2.1 Dirichlet Process

The Dirichlet process is presented as a measure (Section 2.1.1), is shown to have a sequential representation in the form of the Chinese restaurant process (Section 2.1.2), is used as a prior for a mixture (Section 2.1.3), and a stick-breaking representation (Section 2.1.4). We compare the Dirichlet process, the Dirichlet process as prior for a mixture model, the Chinese restaurant process and the stick-breaking representation in Figure 2.1 using plate notation (cf. Fox et al., 2007).

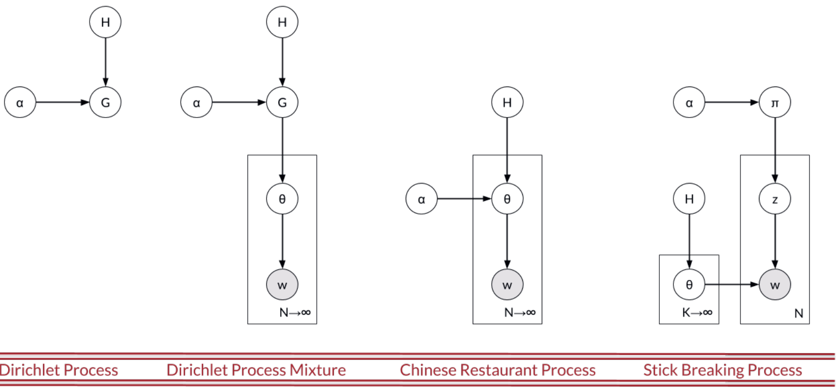

Figure 2.1: From left to right. (1) The Dirichlet process $G \sim DP(\alpha, H)$. (2) The Dirichlet mixture model with G as a prior: $\theta_i|G \sim G$. The parameters θ_i generate observations w_i through $p(w_i|\theta_i)$. (3) The Chinese restaurant process with G marginalized out. (4) The stick-breaking process with a distribution over partition sizes π and indicator variables z_i (cf. Fox et al., 2007).

2.1.1 Dirichlet Process as a Measure

The Dirichlet process (DP) is a distribution over distributions (Ferguson, 1973).

▼ Definition 2.1 — *Dirichlet process*

A **Dirichlet process** DP over a set S can be used to draw sample paths G:

$$G \sim DP(\alpha, H)$$

with α the dispersion parameter and H a measure on S and for which any measurable partition $\{B_0, \dots, B_{n-1}\} \in S$ is drawn from a Dirichlet distribution:

$$(G(B_0), \dots, G(B_{n-1})) \sim \text{Dirichlet}(\alpha H(B_0), \dots, \alpha H(B_{n-1}))$$

The Lévy intensity of the Dirichlet process is complicated, because it is a so-called normalized process, see Regazzini et al. (2003).

2.1.2 Chinese Restaurant Process

De Finetti's theorem (Definition A.45) can be used to establish the existence of an infinitely exchangeable sequence. In the particular case of the Dirichlet process the sequence is an exchangeable *distribution over partitions* and is called the Chinese restaurant process (Aldous, 1985).

▼ Definition 2.2 — *Chinese restaurant process*

A **Chinese restaurant process** is a sequential process that is an exchangeable distribution over partitions:

$$p(z_i = k | z_0, \dots, z_{i-1}) = \begin{cases} \frac{n_k}{\alpha+i} & \text{if } k \leq K_+ \\ \frac{\alpha}{\alpha+i} & \text{if } k > K_+ \end{cases} \tag{2.1}$$

The conditional probability of a cluster assignment z_i (for sample θ_i) given the cluster assignments $z_0, \dots, z_{i-1}$ is proportional to the number of samples n_k assigned to an existing cluster k, or proportional to α for a new cluster. The existing clusters are denoted by $k \leq K_+$ with K_+ the number of clusters.

Instead of sampling cluster assignments, we can generate the samples $\{\theta\}$ from the base distribution H directly:

$$\theta_{i+1} | \theta_1, \dots, \theta_i \sim \begin{cases} \delta_{\theta_1}, & \text{with probability } 1/(\alpha+i), \\ \vdots & \vdots \\ \delta_{\theta_i}, & \text{with probability } 1/(\alpha+i), \\ H, & \text{with probability } \alpha/(\alpha+i). \end{cases} \tag{2.2}$$

The random measure G is marginalized out (see also the plate notation in Figure 2.1). This particular representation is especially convenient for Gibbs sampling about which we will learn in Section 2.2.4. This can also written equivalently as:

$$\theta_{i+1} \mid \theta_1, \dots, \theta_i \sim \frac{1}{\alpha+i}\left(\alpha H + \sum_{j=1}^{i} \delta_{\theta_j}\right). \tag{2.3}$$

The Chinese restaurant process is visualized in Figure 2.2. In a restaurant with K^+ tables there are i customers seated, $\{\theta_0, \dots, \theta_{i-1}\}$. The probability that a new customer, θ_i, will be seated at an existing table depends on the number of customers, n_k, already at the table. The probability that the customer is assigned a new table is proportional to the dispersion parameter α.

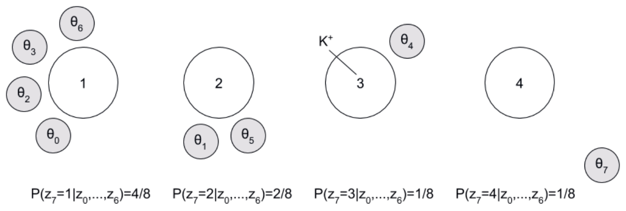

Figure 2.2: The Chinese restaurant process with i customers, $\theta_0, \ldots \theta_{i-1}$, already sitting down. A new customer θ_7 arrives and gets assigned. The assignment variable is z_7. The customer gets assigned an existing table $\{1, 2, 3\}$ with a probability proportional to the number of customers n_k sitting at that table: $n_k/(\alpha + i)$. The customer gets assigned a new, empty table $\{4\}$ with probability $\alpha/(\alpha + i)$. In the visualized Chinese restaurant process the dispersion factor α is equal to 1. Thus for the first table $n_k = 4$ (four customers) and the probability the customer will be assigned this table: $P(z_7 = 1|z_0, \ldots, z_6) = n_k/(\alpha + i) = 4/(1 + 7) = 4/8$.

2.1.3 Dirichlet Process Mixture

The Dirichlet process can be used as a prior for an infinite mixture model (Definition A.42). This is visualized in (Figure 2.3).

	θ0	θ1	θ2	θ3	θ4	θ5	θ6	θ7	θ8	θ9	sum
data 0	0	0	1	0	0	0	0	0	0	...	1
data 1	1	0	0	0	0	0	0	0	0	...	1
data 2	1	0	0	0	0	0	0	0	0	...	1
data 3	0	1	0	0	0	0	0	0	0	...	1
data 4	1	0	0	0	0	0	0	0	0	...	1
data 5	0	1	0	0	0	0	0	0	0	...	1
data 6	0	0	0	0	1	0	0	0	0	...	1

Figure 2.3: An infinite matrix representation of an infinite mixture model. At the horizontal axis we see the latent variables. There are potentially an infinite number of latent variables. At the vertical axis we see the data items. A data item is assigned to a single latent variable. The rows sum up to one.

The representation as a distribution over infinite matrices can be found in the literature (Ghahramani and Griffiths, 2005). We observe that a column has multiple nonzero values. A parameter is not unique, it can be sampled multiple times. The Dirichlet process lends itself as a prior for a mixture. First, the Dirichlet process exhibits this property of being *almost surely discrete* (Ferguson, 1973; Blackwell, 1973; Basu and Tiwari, 1982). If the base measure H is atomless then with probability one the ith value θ_i drawn from the Chinese restaurant process is *distinct* from the values $\theta_1, \ldots, \theta_{i-1}$ if it is drawn from H. Informally, this can be seen as defining a prior over more than one column. Second, for a finite dispersion parameter α, the probability of sampling *identical* values (from $\sum_j \delta_{\theta_j}$) is nonzero. Informally, this can be seen as a prior that allows for multiple nonzero values in a column.

2.1.4 Stick-breaking Representation of the Dirichlet process

The *stick-breaking representation* by Freedman and Diaconis (1983), also known as the residual allocation model (Sawyer and Hartl, 1985; Hoppe, 1986), represents a random process through breaking sticks.[1] A stick of unit length is broken in subsequently smaller pieces with each piece broken off put aside. Here we introduce the stick-breaking representation according to the exposition in Ishwaran and James (2001).

▼ Definition 2.3 — *stick-breaking*

An infinite sequence of random variables $\pi = \{\pi_0, \pi_1, \ldots\}$ has a **stick-breaking representation** with parameters α and β denoted by $\pi \sim GEM(\alpha, \beta)$ with

$$w_k \overset{iid}{\sim} Beta(1-\beta, \alpha + k\beta), \qquad k = 1, \ldots, \tag{2.4}$$

$$\pi_k = w_k \prod_{i=1}^{k-1} (1 - w_i). \tag{2.5}$$

The random variables w_k are drawn iid from a Beta distribution. The stick lengths are represented by π_k and when $k \to \infty$ the infinite dimensional prior is well defined under some mild conditions (Ishwaran and James, 2001). An illustration of the stick-breaking presentation can be found in Appendix A.8. The letters GEM stand for Griffiths, Engen, and McCloskey as proposed - and considered appropriate as acronym because of its beautiful properties - by Ewens (1990).

▼ Definition 2.4 — *stick-breaking representation of the Dirichlet process*

The **stick-breaking representation** of the Dirichlet process states that if

$$\pi_k \sim GEM(\alpha, 0), \qquad k = 1, \ldots, \infty, \tag{2.6}$$

$$\theta_k \overset{iid}{\sim} H, \qquad k = 1, \ldots, \infty, \tag{2.7}$$

$$G = \sum_{k=1}^{\infty} \pi_k \delta_{\theta_k}, \tag{2.8}$$

then $G \sim DP(\alpha, H)$.

[1] Not using the stick analogy but preceding the work by Freedman and Diaconis (1983) and using the same multiplicative representation is the study on how one pound of gold dust can be distributed among a countably infinite number of beggars by Halmos (1944).

The weights π_k are sampled from the stick-breaking process $GEM(\alpha, 0)$. The parameter values θ_k are independently sampled from the base measure H. To sample from the Dirichlet process we sample the parameters θ_k with the weights π_k.

The stick-breaking process can be used as a prior for a mixture. Sample the cluster assignments z_i according to the mixing proportions π and generate the observations from the cluster parameters:

$$z_i \sim Mult(\pi), \tag{2.9}$$
$$w_i \sim F(\theta_{z_i}) \tag{2.10}$$

Here $\theta_k = \theta_{z_i}$ for observation w_i with index i and cluster assignment k: $z_i = k$.

2.2 Inference

There will be six inference methods described, all sampling methods. Inverse transform sampling is described in Section 2.2.1. Rejection sampling in Section 2.2.2. Approximate Bayesian computation in Section 2.2.3. Gibbs sampling in Section 2.2.4. Metropolis-Hastings sampling in Section 2.2.5. Split-Merge MCMC sampling in Section 2.2.6. We report for every inference method the corresponding algorithm in pseudo code. We compare the inference methods in Section 2.2.7.

2.2.1 Inverse Transform Sampling

Let $p(x)$ be a discrete probability distribution with two possible values $x = f$ and $x = g$. The probability distribution sums up to one: $\sum_v p(x = v) = 1$. Sample from a uniform distribution $u \sim U(0,1)$. If $u < p(x = f)$ generate f, else generate g. This procedure samples f with probability $p(x = f)$ and g with probability $p(x = g)$. It can be readily generalized to more than two values by making use of the cumulative distribution function. In Algorithm 1 we sample from $f(x)$ by making use of the inverse cumulative distribution.

Algorithm 1 Inverse transform sampling for $f(x)$

```
1:  procedure INVERSE TRANSFORM SAMPLING($f(x)$)      ▷ Distribution to sample from.
2:      $F(x) = \int_{-\infty}^{x} f(s)ds$             ▷ Create cumulative distribution function $F(x)$.
3:      $X = \varnothing$
4:      for $t = 1 \rightarrow T$ do
5:          $u \sim U(0,1)$                            ▷ Sample from uniform distribution.
6:          $x \sim F^{-1}(u)$                         ▷ Sample $x$ from (the inverse) $F^{-1}(x)$.
7:          $X = X \cup x$
8:      end for
9:      return $X$                                     ▷ $X$ will have the distribution of $f(x)$.
10: end procedure
```

The term “inverse” stems from the fact that we return x (or $f(x)$) given u. Inverse transform sampling is a common component in sampling methods. When one of the steps in an algorithm samples from a uniform distribution, it is often an inverse transform sampling step.

2.2.2 Rejection Sampling

Let $f(x)$ be a complicated function from which it is hard to take samples. Let $g(x)$ be a simple function that is easy to sample from. Then we can sample from $f(x)$ by making sure $Mg(x) \geq f(x)$. The function $Mg(x)$ is an *envelope* function. This sampling method S generates the sample set X using $f(x)$ and $g(x)$.

$$X = S(f(x), g(x), M, T) \tag{2.11}$$

The rejection sampling method (Halperin and Burrows, 1960) for $f(x)$ is described in Algorithm 2.

Algorithm 2 Rejection sampling for $f(x)$

```
 1: procedure REJECTION SAMPLING($f(x), g(x), M, T$)   ▷ Target and proposal distribution
    and scalars M and T.
 2:     $X = \varnothing$
 3:     for $t = 1 \to T$ do
 4:         $x^t \sim g(x)$                              ▷ Generate $x^t$ from $g(x)$
 5:         $u \sim U(0,1)$                              ▷ Inverse transform sampling
 6:         $p_0 = f(x)/(Mg(x))$
 7:         if $u < p_0$ then
 8:             $X = X \cup x^t$                         ▷ Accept
 9:         end if
10:     end for
11:     return $X$                        ▷ $S$ will have the distribution of $f(x)$
12: end procedure
```

We can use rejection sampling to *sample* from the *posterior* $f(\theta|x)$ given that (1) we know the *exact* likelihood function and (2) we can *sample* from the prior. Here, we know that we can sample from the posterior by sampling from $p(\theta)p(x|\theta)$. Moreover, we know that the prior $p(\theta)$ necessarily has to be larger than $p(\theta)p(x|\theta)$ for any observation, because $p(x|\theta)$ is a probability density function, hence for each x and θ it is smaller than one. Therefore we can use rejection sampling with $Mg(x) \geq f(x)$ with $M = 1$, $p(\theta) = g(x)$ and $p(x|\theta = f(x)$.

We introduce the following notation. We make explicit that we need $p(x|\theta)$ for each combination of observations and parameters, but that we only need to *sample*[2] from the prior, which we indicate by a tilde, $\sim p(\theta)$.

[2]The notation $\sim p(\theta)$ means that we can sample from $p(\theta)$ but that we do not have access to a closed-form probability density function.

$$\Theta = S(\sim p(\theta), p(x|\theta), x, T) \tag{2.12}$$

Algorithm 3 Rejection sampling for $f(\theta|x)$

```
1:  procedure REJECTION SAMPLING($p(\theta), p(x|\theta), x$)    ▷ Requires prior, likelihood and observations.
2:      $\Theta = \varnothing$
3:      for $t = 1 \to T$ do
4:          $\theta^t \sim p(\theta)$                              ▷ Generate $\theta^t$ from prior
5:          $u \sim U(0,1)$                                  ▷ Inverse transform sampling
6:          $p_0 = p(x|\theta)$
7:          if $u < p_0$ then
8:              $\Theta = \Theta \cup \theta^t$                              ▷ Accept
9:          end if
10:     end for
11:     return $\Theta$                      ▷ $\Theta$ will have the distribution of $f(\theta|x)$
12: end procedure
```

In Algorithm 3 the envelope distribution $p(\theta)$ and the target distribution $p(\theta)p(x|\theta)$, cancel in such way that only $p(x|\theta)$ remains.

Most examples illustrate rejection sampling by estimating the area of a circle, but let us visualize the method in the context of sampling (Figure 2.4).

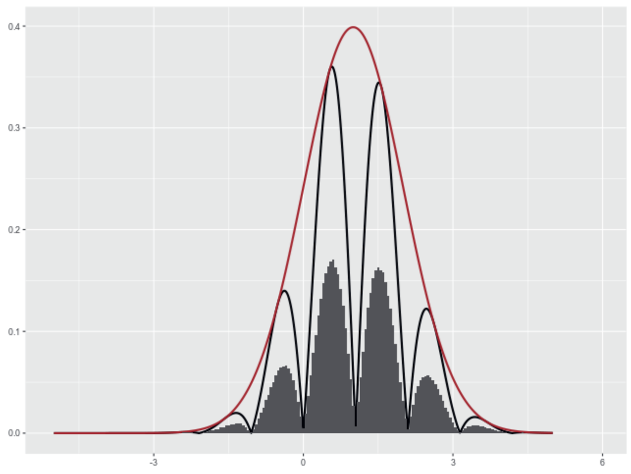

Figure 2.4: A Gaussian is placed over the complex target probability density function. Subsequently the samples that fall in between these two ‘envelopes’ are rejected. This results in a sampling scheme that follows exactly the more complicated probability density function. Note that if the function is scaled by a factor, the sampling scheme stays the same. Such a scaling factor is only important if we want, for example, to know the area under the graph.

2.2.3 Approximate Bayesian Computation

In approximate Bayesian computation (ABC) (Rubin, 1984) the likelihood function does not need to be calculated[3] (Sisson and Fan, 2011). In contrast, it is assumed that there is a model available that simulates observations given the (searched for) parameters. In ABC for each configuration of parameters a set of observations is generated.

$$\Theta = S(\sim p(\theta), X, \sim M(\theta), d(X^t, X), \epsilon, T) \tag{2.13}$$

Approximate Bayesian computation uses many tuning parameters. Its most salient characteristic though, is that it generates pseudo-observations through $M(\theta)$ (see Algorithm 4).

Algorithm 4 Approximate Bayesian computation

```
 1: procedure APPROXIMATE BAYESIAN COMPUTATION($p(\theta), X, M, d, \epsilon$)    ▷ Requires prior,
    observations, model, distance function, and threshold.
 2:     $\Theta = \varnothing$
 3:     for $t = 1 \to T$ do
 4:         $\theta^t \sim p(\theta)$                                   ▷ Generate $\theta$ from prior
 5:         $X^t \sim M(\theta)$                          ▷ Simulate observations $X^t$ from model $M$
 6:         $\rho = d(X^t, X)$   ▷ Calculate distance between simulated and actual observations
 7:         if $\rho \leq \epsilon$ then
 8:             $\Theta = \Theta \cup \theta^t$          ▷ Accept $\theta^t$ if distance falls under threshold $\epsilon$.
 9:         end if
10:     end for
11:     return $\Theta$                          ▷ $\Theta$ will have the distribution of $f(\theta|X)$
12: end procedure
```

The term Bayesian reflects the fact that a prior is involved. The weight of this prior can be manipulated by the threshold ϵ. If this threshold is set very low, the prior plays no role and only observations are taken into account. If ϵ is set extremely high, all θ coming from the prior will be accepted, and the actual observations are not used in the process. There are several disadvantages to approximate Bayesian computation.

- A set of simulated observations has to be compared with the actual observations. This becomes unwieldly if there are many observations.
- It is possible to use summary statistics rather than the observations themselves. If these are sufficient statistics there will be no information loss. If not, there will be information loss in practice.
- The distance function suffers from the curse of dimensionality. In the case that the dimensionality of the individual observations becomes high, or the number of parameters becomes large, it gets increasingly difficult to come up with a distance function which is efficient and accurate at the same time.

[3]ABC is also called likelihood-free computation

2.2.4 Gibbs Sampling

Gibbs sampling (Geman and Geman, 1984) is similar to the *coordinate descent* optimization algorithm (Wright, 2015). In coordinate descent a local minimum of a function is found by iteratively performing a line search along one coordinate direction at a time. Gibbs sampling generates posterior samples by sampling all conditional distributions $p(\theta_i|\theta_{-i})$ iteratively.

$$\Theta = S(\sim p(\theta_i|\theta_{-i}), \sim p(\theta), B, T). \tag{2.14}$$

Sampling iteratively will give us the joint distribution $p(\theta_i, \theta_{-i})$. We can also sample conditional on observations X to obtain $p(\theta|X)$. We will describe Gibbs sampling in that context:

$$\Theta = S(X, \sim p(\theta_i|\theta_{-i}, X), \sim p(\theta), B, T). \tag{2.15}$$

We visualize Gibbs sampling to obtain conditional distributions as in Eq. 2.15 in Figure 2.5.

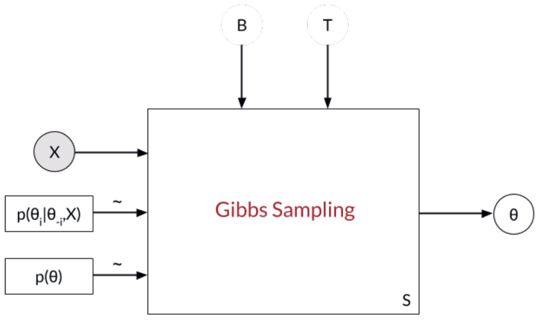

Figure 2.5: A block diagram to visualize Gibbs sampling. This representation makes the inputs and outputs of the Gibbs sampling method explicit. The inputs are observations, X, a conditional distribution to sample from $p(\theta_i, \theta_{-i}, X)$, a prior distribution to sample prior values $p(\theta)$ and a burn-in period, B, and total number of steps, T. The $\sim$ symbol at an arrow indicates sampling rather than access to a probability density function.

The Gibbs algorithm is given in Algorithm 5. Some explanation on the notation is as follows. Here θ can be understood as the parameters of a multivariate probability distribution. The individual parameters are denoted by θ_i. The set of all parameters except for parameter i is denoted by θ_{-i}.

$$\theta_i^t \sim p(\theta_i^t|\theta_{-i}^{t-1}, X). \tag{2.16}$$

The sampling of θ_i^t is here represented in a simplified manner. The actual[4] implementation samples a new parameter value using a combination of already updated, θ^t, and old values, θ^{t-1}:

[4]Also this is simplified. There is a choice to shuffle θ_i^t at each time step or to keep the same order $\theta_1^t, \ldots, \theta_k^t$.

$$\theta_i^t \sim p(\theta_i^t | \theta_1^t, \ldots, \theta_{i-1}^t, \theta_{i+1}^{t-1}, \ldots, \theta_k^{t-1}, X). \tag{2.17}$$

If we sample a parameter we write θ^t with t the iteration or sampling round. The array of parameter samples has capital letter Θ.

Algorithm 5 Gibbs sampling

```
1: procedure GIBBS SAMPLING($p(\theta_i|\theta_{-i},X), p(\theta), X, B$)  ▷ Requires conditional and prior
   distributions (to sample from), observations, and burn-in period.
2:     $\Theta = []$
3:     $\theta^0 \sim p(\theta)$                          ▷ Set parameters to some initial value.
4:     for $t = 1 \to T$ do
5:         for $i = 1 \to k$ do
6:             $\theta_i^t \sim p(\theta_i^t|\theta_{-i}^{t-1}, X)$   ▷ Generate $\theta_i^t$ from Eq. 2.16, equivalently, Eq. 2.17.
7:         end for
8:         $\Theta[t] = \theta^t$
9:     end for
10:    $\Theta_{B:T} = \Theta[B:T]$                ▷ Get samples from burn-in $B$ to end of run $T$.
11:    $\Theta \sim \Theta_{B:T}$                          ▷ Sample $\Theta$ from correlated $\Theta_{B:T}$.
12:    return $\Theta$
13: end procedure
```

Gibbs samples are Markovian. This means that the conditional probability only takes into account values at the previous time step $t-1$. When running the Gibbs sampling algorithm long enough, it will visit all possible states eventually. The Markovian property has an undesired side effect. It makes subsequent steps correlated. Hence, when finally extracting the parameter probabilities, it is important to skip multiple steps to remove the temporal correlations. It is also important to run the algorithm for a while after its start. In that case it does not suffer from a bad choice of initial parameter values. Disregarding the first samples is called burn-in. In words, Gibbs sampling works by having the algorithm spend time in parts of the space proportionally to the probability of getting into that part of the space.

2.2.5 Metropolis-Hastings Sampling

Metropolis-Hastings (Metropolis et al., 1953; Hastings, 1970) is one of the most well-known Markov chain Monte Carlo (MCMC) algorithms. An MCMC algorithm uses a Markov chain (see Gibbs sampling, Section 2.2.4) and combines this with a stochastic (Monte Carlo) component. This sampling method can be used for high-dimensional distributions. Metropolis-Hastings calculates an acceptance factor α which takes into account if a step should be taken according to a predefined proposal distribution. In case this step is not accepted, the current sample is resampled (see Algorithm 6).

$$\Theta = S(X, \theta^0, Q(\theta^t|\theta^{t-1}), f(\theta; X)) \tag{2.18}$$

Metropolis-Hastings can be described without reference to observations just as with Gibbs sampling. However, Eq. 2.18 and Algorithm 6 is described including the observations (which seems appropriate within the Bayesian context). We observe that in contrast to Gibbs sampling, we need to be able to calculate $Q(\theta^t|\theta^{t-1})$, rather than only be able to sample from $p(\theta_i^t|\theta_{-i}^{t-1})$.

Algorithm 6 Metropolis-Hastings sampling

```
 1: procedure METROPOLIS-HASTINGS SAMPLING($\theta^0, X, Q, f$) ▷ Requires initial parameters, observations, proposal distribution, and function proportional to desired distribution
 2:     $\Theta = []$
 3:     for $t = 1 \to T$ do
 4:         $\theta^t \sim Q(\cdot|\theta^{t-1})$                    ▷ Sample from proposal distribution $Q$
 5:         $\alpha = \frac{f(\theta^t;X)Q(\theta^t|\theta^{t-1})}{f(\theta^{t-1};X)Q(\theta^{t-1}|\theta^t)}$                    ▷ Calculate acceptance
 6:         $u \sim U(0,1)$                    ▷ Inverse transform sampling
 7:         if $\alpha < u$ then
 8:             $\theta^t = \theta^{t-1}$          ▷ Reuse previous sample (note, different from rejection)
 9:         end if
10:         $\Theta[t] = \theta^t$                    ▷ Always add $\theta^t$ (accepted or repeated).
11:     end for
12:     return $\Theta$                    ▷ $\Theta$ will be samples from the distribution $f(\theta;X)$
13: end procedure
```

A particular choice of a Metropolis-Hastings step is that of a proposal distribution that does not depend on the state of the chain. This is already suggested by Hastings and is called the independence sampler (Hastings, 1970).

2.2.6 Split-Merge MCMC Sampling

When we study the model as described in Section 2.1.2 we see that samples can be modelled as being clustered. The discussed sampling methods do not assume such kind of structure in the model. This means that in hierarchical models sampling either occurs through updating the to-be-estimated quantities by iterating over every single observation or over every single parameter. This has a disadvantage, an algorithm in which a new cluster is formed by sampling over individual data points can be quite slow. It would be more efficient if multiple data points can be assigned at once to a new parameter. This corresponds to methods in which we can split or merge clusters of data points.

Split-merge samplers are such methods that can update cluster assignments for multiple observations at once. These samples adjust the acceptance method in the Metropolis-Hastings algorithm. Split-Merge sampling is described in Algorithm 7.

Algorithm 7 Split-Merge MCMC sampling

```
1: procedure SPLIT-MERGE MCMC SAMPLING($\theta^0$,$X$,$Q$,$f$)   ▷ Requires initial parameters, observations, proposal distribution, and function proportional to desired distribution
2:     $\Theta = []$
3:     for $t = 1 \rightarrow T$ do
4:         $i \sim U(\{0,\dots,N-1\})$   ▷ Sample observation $i$ discretely.
5:         $j \sim U(\{0,\dots,N-1\} \setminus i)$   ▷ Sample observation $j$ discretely with $j \neq i$.
6:         if $c_i == c_j$ then   ▷ Let us consider a split of this cluster ($c_i = c_j$).
7:             $c_{old} = c_i$
8:             $\theta^t_{c_{new}} \sim Q(\theta^t|\theta^{t-1})$   ▷ Sample from proposal distribution $Q$.
9:             for $k \in c_{old}$ do
10:                $c_k \sim Cat(c_{old}, c_{new})$   ▷ Assign to new cluster categorically.
11:            end for
12:        else   ▷ Let us consider a merge of these clusters ($c_i \neq c_j$).
13:            $c_{merge} = c_i$
14:            for $k \in c_j$ do
15:                $c_k = c_{merge}$   ▷ Assign all observations to the first cluster.
16:            end for
17:        end if
18:        $\alpha = \frac{f(\theta^{t+1},X^{t+1})Q(\theta^{t+1}|\theta^t)}{f(\theta^t,X^t)Q(\theta^t|\theta^{t+1})}$   ▷ Calculate acceptance.
19:        $u \sim U(0,1)$   ▷ Inverse transform sampling.
20:        if $\alpha < u$ then
21:            $\theta^t = \theta^{t-1}$   ▷ Set current sample to previous sample.
22:        end if
23:        $\Theta[t] = \theta^t$   ▷ Always add $\theta^t$ (accepted or repeated).
24:    end for
25:    return $\Theta$   ▷ $\Theta$ will be samples from the distribution $f(\theta|x)$.
26: end procedure
```

The exact acceptance probability depends on the model. For the mixture model with a Dirichlet Process as prior, its performance is further improved by adjusting the assignment process from random to observation-supported by introducing intermediate restricted Gibbs sampling steps (Jain and Neal, 2004, 2007). Similarly, there are other variants that incorporate data fit to the splitting step. Labels can for example be calculated sequentially (Dahl, 2003) or methods can be used that postulate subcluster structure within clusters to optimize inference over split and merge sets (Chang and Fisher III, 2013).

2.2.7 Comparison of the Six Inference Methods

In robotic vision the type of data we are obtaining from depth sensors are point clouds. To perform inference over objects made out of point clouds, clustering algorithms benefit from two sampling strategies. If conjugate probability densities are used, Gibbs sampling, or collapsed Gibbs sampling can be used (Section 2.2.4). If the model consists of a nonconjugate prior and likelihood function, Metropolis-Hastings sampling can be used (Section 2.2.5). If the model becomes more complicated split-merge sampling might accelerate the inference process (Section 2.2.6).

2.3 Chapter Conclusions

In this chapter we introduced the Dirichlet process in Section 2.1. The Dirichlet process is described as a measure (Section 2.1.1), represented as a Chinese restaurant process (Section 2.1.2), described as inducing a mixture (Section 2.1.3), and constructed through stick-breaking (Section 2.1.4). The four expositions serve different purposes. The Dirichlet process as a random measure puts on firm theoretic grounds. The Chinese restaurant process representation lends itself to Gibbs-like sampling methods. The representation as a prior over infinite matrices shows how the Dirichlet process can be used as a prior for a mixture model. The stick-breaking construction shows how the cluster sizes are distributed rather than the individual samples.

In Section 2.2 six inference methods are described. Inverse transform sampling, rejection sampling, approximate Bayesian computation, Gibbs sampling, Metropolis-Hastings, and Split-Merge Markov chain Monte Carlo. These methods use different types of information. Inverse transform sampling requires the cumulative distribution function (or more specific its inverse). Rejection sampling requires an envelope function that approximates the actual distribution sufficiently well, such that not many samples will be rejected. If the likelihood function is known, and samples can be drawn from the prior, the likelihood function can be used to sample from the posterior. Approximate Bayesian computation does not require a likelihood function, but manipulates the influence of the prior by a threshold and introduces a distance function to define if simulated and actual observations are close. Gibbs sampling does not use such artificial parameters, but requires a prior and likelihood that are conjugate. It can be used for models where this is the case. Metropolis-Hastings sampling is an MCMC sampling method that can be used for nonconjugate models. Its convergence rate depends on the quality of a proposal distribution. To accelerate convergence, split-merge MCMC sampling performs inference not just over individual data points, but over sets of data points (e.g., splitting a cluster or merging two clusters of data).

2.4 A Coda

Below we provide a critical confrontation of our conclusions in relation with existing mathematical theories. First of all, we would like to stress that this research work is of a technical and applied nature as the title implies: *Nonparametric Bayesian Methods in Robotic Vision*. Below we discuss some theoretical research directions that we have neither explored in this chapter, nor in the following chapters.

Robustness is an important aspect of Bayesian methods (Ghosal and Van der Vaart, 2017). The choice of prior should not influence the posterior distribution "too much". This is difficult to study on its own, for which reason *posterior consistency* is studied instead. It loosely means that the posterior probability is eventually concentrated in a small neighborhood of the "actual value" of the parameter. The study of asymptotic properties, such as posterior consistency, is more complex in the nonparametric case. An infinite amount of data might not overcome the prior. In this thesis we rely on the use of a Dirichlet process prior for robotic (depth) vision tasks. There will neither be an analysis on the consistency of the prior, nor on the rate of contraction as advocated for by Ghosal and Van der Vaart (2017).

One of the reasons to introduce the class of Dirichlet priors is to ensure consistency in the nonparametric Bayesian setting (Diaconis and Freedman, 1986). However, it is possible to use them in such a way that they lead to inconsistent estimates. The authors describe a Dirichlet prior with a Cauchy distribution as base measure crafted in such way that it does not converge in the asymptotic case to the underlying true distribution that consists of two point masses positioned at locations $-a$ and a with respect to the origin. This means that the posterior might converge to the wrong value or that the estimates will oscillate in the asymptotic case. We will not explore possible inconsistent behavior of the base measures that we introduce in the future chapters.

The robotic vision applications do have practical constraints which are apparent from the domain. For example, in Chapter 4 the Pareto prior will not "wash out" if chosen too large. The method will not find segments if they are much smaller than the experimenter expects. There are also degenerate cases such as single outliers or line segments aligned head-to-tail forming a larger, single line segment. Facing the existing literature we observe the following.

- Our practical application will be clustering. We assign points to geometric objects. Lack in convergence towards object parameters does not imply an asymptotic error in the assignment problem.
- In practical situations, there are degenerate cases, (1) outliers, (2) accidentally aligned line segments, and (3) other types of errors such as perceptual resolution, that would require our attention in improving our clustering results.
- Theoretically, even if we do not have the "correct model", we might still come close to it in a precise sense (see Ghosal and Van der Vaart (2017) on Kullback-Leibler projections).

In the next chapters, we will refrain from going into theory. We will only apply existing theory and implement methods to perform robotic (depth) vision in general and perform inference on point clouds in particular.