Dominik Joho

Learning and Utilizing Spatial Object Relations for Service Robots

Learning and Utilizing Spatial Object Relations for Service Robots

Dominik Joho

Dissertation zur Erlangung des akademischen Grades Doktor der Naturwissenschaften
Technische Fakultät, Albert-Ludwigs-Universität Freiburg

Dekan	Prof. Dr. Yiannos Manoli
Erstgutachter	Prof. Dr. Wolfram Burgard
	Albert-Ludwigs-Universität Freiburg
Zweitgutachter	Prof. Dr. Bernhard Nebel
	Albert-Ludwigs-Universität Freiburg
Tag der Disputation	10. Dezember 2013

Abstract

Service robots that operate in domestic environments and assist in the daily housework are still beyond reach when considering the current state of the art. Hence, they pose many interesting technical challenges that currently motivate a considerable amount of basic research in mobile robotics. One of these challenges is the question how service robots could attain the required level of autonomy and reliability to be truly useful. Ideally, if a service robot is deployed in a domestic environment it should require only a short setup time and as few user interactions as possible. When tidying up, it should know where the objects usually belong to. If the robot should set the table, it needs to know which objects are required, where to find them, and how they should be arranged on the table. It is apparent that most of these tasks involve knowledge about the spatial context of objects and how objects are usually arranged in such environments. It would be desirable to have a flexible approach in which the robot adapts to the specifics of its environment by observing it. It thereby could learn the usual object arrangements. Hence, as a basic requirement for achieving a high level of autonomy, service robots need to represent, learn, and utilize knowledge about the relevant spatial relations between objects in man-made environments. If robots are able to utilize representations that take into account the interdependencies between objects in such environments then this would enable them to more efficiently carry out their tasks or to address completely new tasks. For example, robots could more efficiently search for objects, or a robot could reason about missing or misplaced objects in a room or on a table.

In this thesis, we propose several techniques for learning and utilizing spatial object relations. First, we consider the problem of localizing objects. As future household objects and retail products might be equipped with an RFID tag, we first present a technique to localize RFID tags. Further, we show that same technique can also be applied to the complementary situation in which we want to localize a mobile robot based on a known map of RFID tag locations. Given that we can localize objects, we move on to address the question of how a robot can efficiently search for objects in an unknown environment. For this, we present two techniques that both aim at speeding up the search process by taking advantage of background knowledge about usual object arrangements acquired in previously seen, similarly structured environments. Specifically, we are interested in efficiently finding a product in an unknown supermarket. The main idea is to exploit knowledge about the co-occurrence of objects to focus on searching the promising regions first and to postpone the non-promising regions. While both approaches utilize learning techniques to leverage the information of previously seen environments, they both rely on predefined spatial relations, like an object being “in the same aisle” as another object. This motivates the final part of this thesis, in which we aim at learning stable spatial relations between objects. More specifically, we wish to learn spatially coherent object constellations in an unsupervised manner from complex multi-object scenes. As an application scenario, we consider tabletop scenes in which the object constellations correspond to place covers. For this, we propose a novel hierarchical nonparametric Bayesian model that represents a prior distribution over scene structures in terms of object constellations. For posterior inference in this model, we present an efficient Markov chain Monte Carlo (MCMC) sampler. By basing our model on the Dirichlet process and the beta-Bernoulli process, the number of object constellations in our model is not fixed. This has practical benefits in the context of lifelong learning, as the robot is able to recognize and integrate previously unseen object constellations into its model in an open-ended fashion and within a single coherent probabilistic framework.

Zusammenfassung

Serviceroboter, die bei der täglichen Hausarbeit helfen, die das Geschirr abwaschen, einkaufen, aufräumen und den Tisch decken, sind noch weit jenseits der heutigen technischen Möglichkeiten. Sie stellen damit aber auch eine interessante technische Herausforderung dar und motivieren derzeit einen guten Anteil der Grundlagenforschung in der mobilen Robotik. Eine jener Herausforderungen ist die Frage, wie Serviceroboter das notwendige Maß an Autonomie und Verlässlichkeit erreichen können, um wirklich nützlich zu sein. Idealerweise sollte ein Roboter, sobald er in einer neuen Umgebung eingesetzt wird, eine möglichst geringe Einrichtezeit benötigen und dabei auf so wenig Benutzerinteraktionen wie möglich angewiesen sein. Das heißt, um das Geschirr zu waschen, sollte er das Geschirr und die Küche selbständig finden. Um aufzuräumen, muss er wissen, wohin die Gegenstände gehören. Um den Tisch zu decken, muss er wissen, welche Gegenstände benötigt werden, wo sie zu finden sind, und wie sie auf dem Tisch zu arrangieren sind.

Es ist offensichtlich, dass die meisten dieser Aufgaben Vorwissen über den räumlichen Kontext von Objekten erfordert. Zum Beispiel muss klar sein, wie Objekte in solchen Umgebungen normalerweise arrangiert sind. Der Benutzer könnte dem Roboter in einer Einarbeitungsphase dieses Wissen vermitteln, oder der Roboter wird mit einprogrammierten Vorwissen ausgeliefert, das es ihm z. B. ermöglicht, einen Tisch zu decken. Beide Möglichkeiten erscheinen jedoch unbefriedigend. Während eine Einarbeitungsphase zeitintensiv sein kann, ist einprogrammiertes Vorwissen unflexibel, wenn es sich nicht an die Wünsche des Nutzers anpassen lässt. Es wäre daher erstrebenswert, einen flexibleren Ansatz zu verfolgen, bei dem der Roboter sich an die Spezifika seiner Umgebung selbst anpasst. Dadurch wäre es ihm möglich, die Objektarrangements und relevanten Objektrelationen selbst durch Beobachtung seiner Umgebung zu lernen.

Als Grundvoraussetzung um wirklich nützlich zu sein und einen hohen Grad an Autonomie zu erreichen, sollte ein Serviceroboter daher Vorwissen über die relevanten räumlichen Objektrelationen repräsentieren, erlernen und einsetzen können. Wenn Roboter in der Lage wären, auf Repräsentationen zurückzugreifen, welche die wechselseitigen Abhängigkeiten zwischen Objekten berücksichtigen, würde sie dies dazu befähigen, ihre Aufgaben effizienter auszuführen, oder komplett neuartige Aufgaben zu erledigen. Beispielsweise könnte ein Roboter effizienter nach Objekten suchen, indem er sich zunächst auf die vielversprechen-

den Regionen konzentriert. Ob eine Region dabei als vielversprechend angesehen wird, sollte hauptsächlich davon abhängen, welche anderen Gegenstände dort bereits erkannt wurden, sowie der Wahrscheinlichkeit, dass der gesuchte Gegenstand zusammen mit diesen Gegenständen auftritt. Zudem könnte ein Roboter mittels solcher Repräsentationen fehlende oder deplatzierte Objekte in einem Raum oder auf einem Tisch erkennen. So könnte der Roboter über die Zeit eine Repräsentation der Gedecke auf einem Frühstückstisch gelernt haben. Dies würde es ihm ermöglichen, die Struktur eines teilweise gedeckten Tisches zu analysieren, die fehlenden Objekte zu identifizieren und dem Menschen beim Decken des Tisches zu assistieren.

In der vorliegenden Dissertation präsentieren wir verschiedene Techniken als erste Schritte in Richtung des langfristigen Ziels der Realisierung eines autonomen Serviceroboters. Zunächst sollte der Roboter in der Lage sein, Objekte zu erkennen. Da zukünftige Haushaltsgegenstände und Produkte bereits mit einem RFID-Etikett versehen sein könnten, schlagen wir eine Methode zur Lokalisierung dieser Objekte auf Basis der RFID-Technik vor. Um die Position der RFID-Etiketten zu bestimmen, verwenden wir eine RFID-Antenne, die auf einer mobilen Sensorplattform oder einem mobilen Roboter montiert ist. Mittels dieser Antenne kann die eindeutige ID der RFID-Etiketten ausgelesen werden. Zudem wird dabei jeweils die Signalstärke gemessen, mit der diese Information empfangen wurde. Der von uns vorgeschlagene Ansatz ist in der Lage, diese Messungen zu einer Wahrscheinlichkeitsverteilung über die Position eines Etiketts zu integrieren. Zudem ist unser Ansatz für die komplementäre Situation einsetzbar, in der die Positionen der RFID-Etiketten bekannt sind und auf Basis der Messungen die Trajektorie des Roboters geschätzt werden soll. In beiden Situationen ist unser Ansatz jedoch auf ein probabilistisches Sensormodell angewiesen, das einen Bezug zwischen der Roboter- und Etikettposition und der erhaltenen Messung herstellt. Ein Hauptbeitrag unseres Ansatzes ist daher ein neuartiges probabilistisches Sensormodell für die RFID-basierte Lokalisierung, das im Gegensatz zu früheren Ansätzen sowohl die empfangene Signalstärke berücksichtigt, als auch die Detektionswahrscheinlichkeit modelliert. In unseren Experimenten zeigen wir, dass dies eine genauere Positionsschätzung erlaubt, als dies durch ein Sensormodell möglich wäre, das nur die Signalstärke oder nur Detektionsereignisse berücksichtigt. Der zusätzlich zu veranschlagende Berechnungsaufwand, der benötigt wird, um beide Modalitäten zu berücksichtigen, ist vernachlässigbar. Das von uns vorgeschlagene Sensormodell fällt in die Kategorie der antennenzentrischen Sensormodelle, welche die erwarteten Messungen in einem Bezugssystem relativ zur Antenne modellieren. Die Kalibrierung solcher Modelle kann recht aufwendig sein. Es müssen Referenzmessungen an verschiedenen antennenrelativen Positionen vorgenommen werden. Dazu kann einerseits die Antenne fix gehalten werden und von einem RFID-Etikett an verschiedenen relativen Positionen Messungen vorgenommen werden. Andererseits könnte man verschiedene Etiketten in der Umgebung verteilen und eine lokalisierte Antenne durch diese Umgebung bewegen. Da sowohl Antennenposition als auch Etikettpositionen bekannt sind, können die Messungen in ein antennenzentrisches Bezugssystem zurückgerechnet werden. Ein großer Nachteil beider Methoden ist jedoch, dass die Positionen der RFID-Etiketten bekannt sein müssen. Wir stellen deshalb ein Verfahren vor, das diese Kalibrierungsphase dahingehend vereinfacht, dass im Vorfeld keine Etikettpositionen ermittelt werden müssen. Dies wird durch ein iteratives Verfahren erreicht, das sowohl das Sensormodell als auch die Etikettpositionen schätzt. In Experimenten zeigen wir, dass das so gelernte Sensormodell gegen ein nicht-iterativ gelerntes Sensormodell konvergiert (bis auf einen gewissen empirischen Fehler). Dabei ist das nicht-iterative Verfahren weiterhin auf bekannte Etikettpositionen angewiesen. Zudem zeigen unsere Ergebnisse, dass die im iterativen Verfahren simultan geschätzten Etikettpositionen gegen die tatsächlichen Positionen konvergieren (bis auf einen gewissen empirischen Fehler). Wir möchten betonen, dass das iterative Verfahren nur notwendig ist, sofern die Annahme fallen gelassen werden soll, dass die Etikettpositionen bekannt seien. Wir werden unser Verfahren und Varianten davon quantitativ in einem Büro und einem Supermarkt evaluieren. Dies schließt einen Vergleich mit aus der Literatur bekannten Methoden ein. Dazu implementieren wir eine WiFi-basierte Lokalisierungsmethode, die eine 2D-Signalstärkekarte mittels Gauß-Prozess-Regression schätzt. Zudem adaptieren wir das Modell, sodass die Signalstärke auch im Posenraum kartiert werden kann. Dies ist besonders für die RFID-basierte Lokalisierung relevant, da die empfangene Signalstärke in erheblichen Maß von der Orientierung der Antenne abhängt.

Im Folgenden wollen wir nun annehmen, dass der Roboter in der Lage ist, Objekte zu lokalisieren. Wir wenden uns dann der Frage zu, wie er Hintergrundwissen über gebräuchliche Objektarrangements ausnutzen kann, um seine Aufgaben effizienter auszuführen, insbesondere, wenn er Objekte in einer unbekannten Umgebung sucht. Zur Motivation veranschaulichen wir uns die Situation, dass wir ein bestimmtes Produkt in einem unbekannten Supermarkt suchen. Sicher würden wir nicht einfach zufällig durch den Markt gehen, aber genauso wenig würden wir systematisch jeden einzelnen Gang ablaufen bis wir das Produkt finden. Vielmehr würden wir unser Suchverhalten von unseren aktuellen Beobachtungen abhängig machen, sowie von unserem Vorwissen darüber, wie Objekte in solchen Umgebungen üblicherweise angeordnet sind. Dieses Vorwissen basiert auf unseren Erfahrungen in anderen Supermärkten, in denen uns gewisse stabile räumliche Abhängigkeiten zwischen Produktgruppen aufgefallen sind. Die Aufgabe ist nun, dieses Wissen zunächst in einer geeigneten Weise zu formalisieren, sodass ein Roboter dies ebenfalls auf Basis von Daten echter Supermärkte erlernen kann. Zudem müssen wir eine Suchstrategie entwerfen, die dieses Wissen für eine effiziente Suche ausnutzen kann.

Wir stellen dafür zwei alternative Ansätze vor. Unser erster Ansatz ist eine reaktive Suchstrategie, welche die Entscheidung, wo als nächstes gesucht werden soll, von den aktuell direkt sichtbaren Objekten der näheren Umgebung abhängig macht. Als Suchheuristik verwendet der Roboter dabei Entscheidungsbäume, welche die Alternativen an einer Weggabelung in einem Supermarkt in vielversprechende und weniger aussichtsreiche Richtungen klassifiziert. Die Entscheidungsbäume werden auf Daten von optimalen Suchpfaden in Trainingsmärkten gelernt. Im Gegensatz dazu verfolgt unsere zweite Suchstrategie einen globaleren, inferenzbasierten Ansatz, der alle bisher gesehenen Objekte und Strukturen im Supermarkt berücksichtigt. Auf Basis dieser Beobachtungen wird eine Verteilung über die mögliche Position des Produkts berechnet. Der Roboter wählt dann einen Zielpunkt aus, indem er die Distanz zu diesem Zielpunkt mit der Wahrscheinlichkeit abwägt, das Produkt an diesem Ort zu finden. Er führt dann seine Suche fort, indem er sich auf dem kürzesten Weg zu diesem Zielpunkt begibt. Sobald neue Informationen zur Verfügung stehen, d.h. neue Produkte oder Strukturen gesehen werden, wird die Verteilung neu berechnet und ein neuer Zielpunkt ausgewählt. Die Verteilung basiert auf einem Modell, das im Grunde die Wahrscheinlichkeit berücksichtigt, dass zwei Objekte in bestimmten räumlichen Kontexten gemeinsam auftreten. Die Parameter dieses Modells können auf Basis von Karten von Supermärkten gelernt werden. Zwar unterscheiden sich beide Suchstrategien in ihren technischen Details, sie werden jedoch beide auf einer Trainingsmenge von drei Märkten trainiert und in einem vierten Supermarkt evaluiert. Die benötigten Daten wurden in vier echten Supermärkten erhoben und modellieren im Detail die Regalaufstellungen und die Platzierung der Produkte. Die Effizienz beider Suchstrategien vergleichen wir quantitativ mit einer Basisstrategie, die den Markt erkundet bis das Produkt gefunden wird. Zudem präsentieren wir einen Vergleich zum Abschneiden von Versuchspersonen die in einer Feldstudie in einem echten Supermarkt nach den gleichen Produkten suchen mussten.

Die inferenzbasierte Suchstrategie nutzt Kookkurrenz-Statistiken über Objekte in verschiedenen räumlichen Kontexten. Z. B. wird die Wahrscheinlichkeit betrachtet, dass das gesuchte Produkt sich “im gleichen Regal” oder “in einem benachbarten Regal” befinden könnte, wie ein anderes, während der Suche bereits gesehenes Produkt. Zwar können die Parameter des Modells auf Basis von Karten von Supermärkten gelernt werden, die dabei betrachteten räumlichen Relationen sind jedoch vordefiniert und werden nicht gelernt.

Deshalb wenden wir uns in unserem letzten Beitrag dieser Arbeit einem allgemeineren Lernszenario zu, in dem wir den räumlichen Kontext von Objekten lernen wollen. Konkret wollen wir unüberwacht räumlich stabile Objektkonstellationen in komplexen Alltagsszenen lernen. Als Anwendungsszenario betrachten wir Tischszenen in denen die relevanten Objektkonstellationen den Gedecken entsprechen, die, z. B., aus den Objekten Teller, Messer und Kaffeetasse bestehen. Die Identifikation von relevanten Objektkonstellationen kann als Parsen der Szene oder als Inferenz der unbekannten Szenenstruktur angesehen werden. Für eine gegebene Szene existieren jedoch im Allgemeinen verschiedene mögliche Interpretationen ihrer Struktur. Eine Szene kann als Ansammlung zufällig verteilter Objekte angesehen werden, oder sie kann als ganzes als eine einzige, große Objektkonstellation interpretiert werden. Für die Szene eines Frühstückstisches, der für drei Personen gedeckt ist, würden wir es jedoch als viel wahrscheinlicher ansehen, dass drei sich wiederholende Objektkonstellationen zu finden sind – eben die Gedecke. Wir werden den Begriff einer "wahrscheinlicheren Szenenstruktur" präzisieren, indem wir eine a-priori-Verteilung über Szenenstrukturen definieren. Diese Verteilung könnte dann zur Evaluation der Wahrscheinlichkeiten der oben genannten alternativen Szenenbeschreibungen herangezogen werden. Zudem kann diese Verteilung aktualisiert werden und somit eine a-posteriori-Verteilung über Szenenstrukturen modellieren, die Informationen über bereits analysierte Szenen miteinbezieht. Wenn ein Roboter noch nie eine Frühstücksszene gesehen hat, könnte er die drei genannten alternativen Szenenstrukturen als mehr oder weniger gleich wahrscheinlich ansehen. Hätte er jedoch bereits mehrere Szenen analysiert, in denen ähnliche Objektkonstellationen auftreten, würde er eher mit unserer Intuition übereinstimmen, dass ein Frühstückstisch für drei Personen drei relevante Konstellationen enthält die gerade den Gedecken entsprechen.

Konkret besteht unser Beitrag in der Definition eines neuen, hierarchischen, nicht-parametrischen Bayes'schen Modells für komplexe Szenen die aus mehreren Objekten aufgebaut sind. Ein grundlegender Baustein unseres Modells sind sogenannte Meta-Objekte, die eine Verteilung über Objektkonstellationen definieren. Wir nehmen an, dass die Objektkonstellationen auf einem Tisch aus diesen Meta-Objekten gesampelt wurden. Meta-Objekte sind als probabilistische, teilbasierte Modelle definiert und besitzen damit eine interne Struktur. Ein "Teil" eines solchen Modells, beinhaltet (a) eine räumliche Verteilung, welche die relative Position des Objekts festlegt, (b) eine kategoriale Verteilung, welche den Typ des Objekts festlegt, das auf diese relative Position gestellt werden soll (Teller, Tasse, etc.) und (c) eine Aktivierungswahrscheinlichkeit, welche festlegt, ob überhaupt ein Objekt auf diese Position zu stellen ist. Ein Meta-Objekt in einer Frühstücksszene könnte z. B. aus vier Teilen bestehen: einem “zentralen Teil”, mit einer hohen Wahrscheinlichkeit für Teller, einem “linken Teil”, mit einer hohen Wahrscheinlichkeit für Gabeln, etc. Um eine Objektkonstellation aus diesem Modell zu sampeln, wird zunächst die Aktivierung der Teile gesampelt. Für jeden aktivierten Teil sampeln wir die relative Position und den Objekttyp. Dadurch generiert jeder Teil eines solchen Modells höchstens ein Objekt. Die Objektkonstellation wird dann in die Szene transformiert, indem wir aus einer a-priori-Verteilung über Transformationen sampeln. Wir nehmen an, dass eine Szene aus mehreren Objektkonstellationen bestehen kann. Zudem nehmen wir an, dass verschiedene *Kategorien* von Konstellationen existieren, mit jeweils unterschiedlichen Parametrisierungen für die Wahrscheinlichkeitsverteilungen ihrer Teile. So könnte eine bestimmte Meta-Objektkategorie mit hoher Wahrscheinlichkeit eine Müslischale und einen Löffel sampeln, während eine andere Kategorie eher Konstellationen mit Teller und Messer sampelt. Eine Szene zu analysieren, heißt, diesen generativen Prozess umzukehren, indem die Anzahl der Objektkonstellationen, deren jeweilige Kategorie und ihre zugehörige Transformation oder Referenzrahmen inferiert werden. Zudem muss jedes Objekt zu einer bestimmten Meta-Objektinstanz assoziiert werden, sowie zu einem bestimmten Teil dieser Instanz. So sollte etwa ein Teller mit dem zentralen Teil eines Meta-Objekts assoziiert werden.

Bisher haben wir noch nicht die *Anzahl* an *Kategorien* von Meta-Objekten oder die Anzahl der *Teile* pro Meta-Objekt-Kategorie in unserem Modell festgelegt. Es wäre wünschenswert, eine genaue Spezifikation dieser Zahlen zu vermeiden, und sie stattdessen aus den Daten zu schätzen. Dies erreichen wir dadurch, dass unser Modell als nicht-parametrisches Bayes’sches Modell auf Basis des Dirichlet-Prozesses und des Beta-Bernoulli-Prozesses definiert wird. Hiermit vermeiden wir sogar, eine Obergrenze für diese Zahlen angeben zu müssen. Stattdessen entspricht dies der Annahme, dass es unendlich viele Kategorien von Meta-Objekten gibt, die jeweils unendlich viele Teile besitzen. Dies bedeutet, dass nun die effektive Anzahl der Meta-Objekt-Kategorien und -Teile ebenfalls aus den Daten inferiert wird. In unserem Modell können wir nun gewisse Hyperparameter setzen, die unsere a-priori Vorstellungen über diese Zahlen widerspiegeln. Dies eröffnet die Möglichkeit, aus den Daten die effektive Modellkomplexität zu inferieren, die sich somit der Datenkomplexität anpassen kann. Dies hat für den Roboter auch praktische Konsequenzen für Szenarien des lebenslangen Lernens. Da die Anzahl der Objektkonstellationen in unserem Modell variabel gehalten ist, kann der Roboter neue, bisher noch nie gesehene Objektkonstellationen als solche erkennen und in sein Modell integrieren – und dies innerhalb eines einzigen, kohärenten probabilistischen Rahmens.

Acknowledgements

A doctoral thesis is the work of a single one, but it is well understood that such a project is impossible without the guidance and support of colleagues and friends. I am indebted to a few people and I want to take the opportunity to thank them here.

First of all, I want to thank Prof. Dr. Wolfram Burgard for giving me the opportunity to work in his group. He gave me considerable freedom to pursue my own interests and I am grateful for this and the trust that this implies. The mindset to approach technical problems from a probabilistic point of view is considered by me to be the most important lesson that I have learned in these years and I attribute this mainly to his influence.

I also want to thank Prof. Dr. Bernhard Nebel for acting as a second referee.

The stimulating and motivating atmosphere at the AIS lab certainly had a great influence on this thesis. There are now so many people working in the lab that I refrain from naming everyone individually. Each one contributed his or her share in creating this atmosphere and every one deserves my gratitude for that. And at times, the AIS lab was more than just a workplace. I especially remember our stay in Kōbe and Ōsaka during ICRA'09 as an outstanding experience.

I really enjoyed the inspiring discussions with Dr. Gian Diego Tipaldi during our work on the scene analysis paper. He supported me in trying out my own ideas, but he also took care to point out to me the things that might not work – and this is perhaps one of the best ways of supervising someone. Especially, he convinced me that my favorite toy from the last paper would likely be of no use for the current problem. So I started looking for a new toy. And I think this was a crucial step.

Further, I want to thank Dr. Christian Plagemann for the collaboration during our work on the RFID paper. He helped me to understand the idea of Gaussian process regression and he also kindly gave me access to his own implementation for Gaussian process regression.

I want to thank Dagmar Sonntag and Susanne Bourjaillat for administrative support, and Michael Keser for technical support. I also thank Martin Senk for the collaboration during his Bachelor's thesis, especially, for the additional work he has done for the conference and journal paper. Further, I thank Nikolas Engelhard for implementing the segmentation and classification of the point clouds for the scene analysis paper.

I also collaborated with Christopher Kalff when we used the RFID localization technique to track the participants in a field study that he conducted in a supermarket. In turn, he provided me with an evaluation of the resulting trajectories, which enabled me to compare the performance of my search technique to the performance of the human subjects (unfortunately, the humans won). I think, this was an interesting addition to one of my papers. Sadly, it is too late now to thank him for this collaboration at this point.

I thank Dr. Barbara Frank, Dr. Zeno Gantner, Markus Kuderer, PD Dr. Cyrill Stachniss, Dr. Gian Diego Tipaldi, and Dr. Thorsten Zitterell for reading parts of an earlier version of this thesis.

Finally and most importantly, I want to thank my wife Karla Alcázar for the support and love throughout these years. She always believed in me and this thesis would not have been possible without her. And hats off to our son Julian, for his proficiency in destroying any meaningful object arrangement in our very own domestic environment on a daily basis.

This work has been supported by the Deutsche Forschungsgemeinschaft (DFG) under contract number SFB/TR 8 Spatial Cognition (R6-[SpaceGuide]).

Para Karla y Julian.

Contents

CHAPTER 1

Introduction

Service robots that operate in domestic environments and assist in the daily housework, that wash the dishes, go shopping, tidy up, and set the table, are still beyond reach when considering the current state of the art. Hence, they pose many interesting technical challenges that currently motivate a considerable amount of basic research in mobile robotics. One of these challenges is the question how service robots could attain the required level of autonomy and reliability to be truly useful. Ideally, if a service robot is deployed in a domestic environment it should require only a short setup time and as few user interactions as possible. That is, for washing the dishes it should find and recognize the plates and the kitchen sink by itself instead of being manually instructed. When tidying up, it should know where the objects usually belong to. If the robot should set the table, it needs to know which objects are required, where to find them, and how they should be arranged on the table.

It is apparent that most of these tasks involve knowledge about the spatial context of objects and how objects are usually arranged in such environments. Hence, the user could either go through a lengthy instruction phase in which he or she directly demonstrates these tasks to the robot, or the robot could have pre-programmed knowledge about how to, for example, set a table. Both options seem unsatisfactory as the first approach is time-consuming and the latter ignores the preferences of the user. Therefore, it would be desirable to have a more flexible approach in which the robot adapts to the specifics of its environment by observing it. Thereby, it could learn the usual object arrangements and relevant spatial relations between objects.

Thus, as a basic requirement for being truly useful and for achieving a high level of autonomy, service robots need to represent, learn, and utilize knowledge about the relevant spatial relations between objects in man-made environments. If robots are able to utilize representations that take into account the interdependencies between objects in man-made environments then this would enable them to more efficiently carry out their tasks or to address completely new tasks that would have been impossible without such representations. For example, robots could more efficiently search for objects by first searching the promising regions and postponing the non-promising regions. If a regions looks promising or not will mainly depend on the objects the robots sees there as well as its prior knowledge about how likely the sought object co-occurs with these objects. Further, a robot could reason about missing or misplaced objects in a room or on a table. For example, over time the robot might have learned a representation of the layout of covers on a breakfast table. Based on this, it could parse a partially laid table, identify the missing objects, and aid the human with setting the table.

In this thesis, we present several techniques as preliminary steps towards the long-term goal of building autonomous service robots. First, a robot obviously must be able to detect and localize the objects in the first place. As future household objects and retail products might already be equipped with RFID tags, we therefore present a novel approach to RFID-based localization and mapping. The location of RFID tags can be estimated by means of an RFID antenna attached to a localized sensor platform or robot. While moving through the environment, the robot collects measurements of the RFID tags that include the signal strength as well as the unique ID of a tag. The measurements are then integrated to compute a distribution over the location of each RFID tag. Further, the approach can also be used for the complementary situation in which we are given a map of RFID tag locations and the task is to estimate the trajectory of a robot moving through this environment. Both of these tasks rely on a sensor model that relates the robot and tag locations to the measurements. Hence, we propose a novel probabilistic sensor model that, in contrast to previously published approaches, explicitly considers both tag detection events as well as the received signal strength in a combined model. Our experiments show that this leads to an improved localization accuracy when compared to sensor models that utilize either only the signal strength or only tag detection events. The additional computational overhead for considering both is negligible. Our proposed sensor model belongs to the class of antenna-centric sensor models, which represent the expected measurements in an antenna-relative frame of reference. The calibration phase for such models can be quite tedious. One needs to obtain reference measurements at several antenna-relative positions. This can be done by either keeping the antenna fixed and placing an RFID tag at different relative locations while recording the sensor measurements. Another option is to attach several tags in the

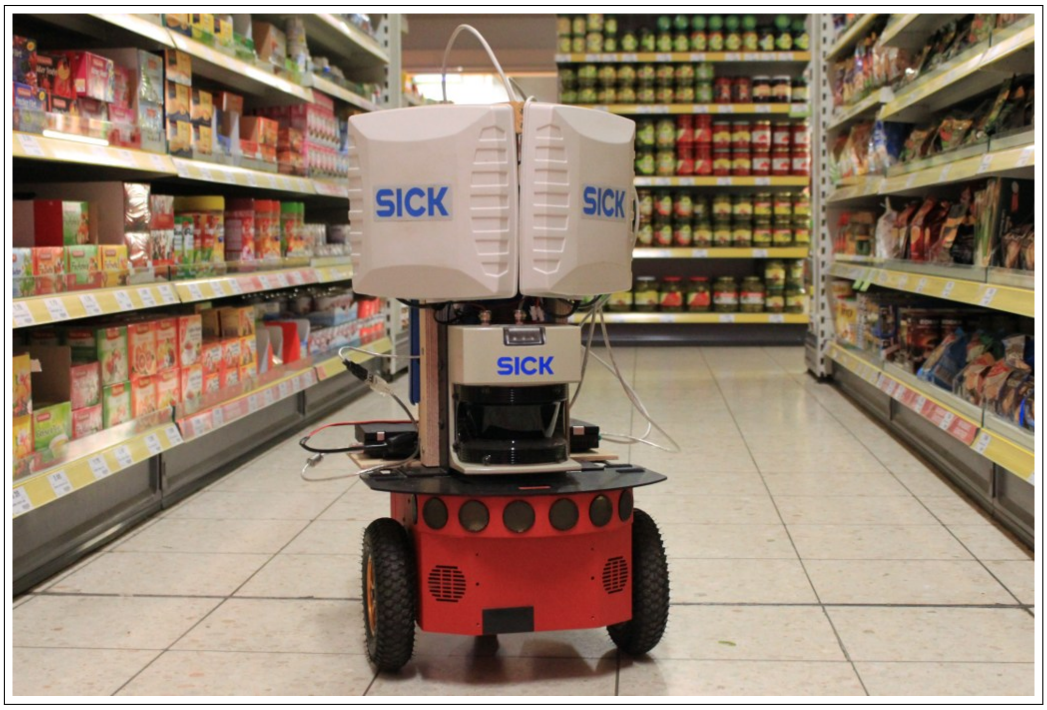


Figure 1.1: A first contribution of this thesis is a technique for object localization. As future household objects and retail products may already be equipped with RFID tags, we propose a novel approach to RFID-based localization and mapping.

environment and to move a localized antenna through the environment. Given that the antenna is localized and the tag locations are known, one can transform the tag locations for each point in time into an antenna-centric reference frame for registering calibration measurements at antenna-relative locations. However, a major drawback is that in either case both the antenna location and the tag locations need to be known. Assuming knowledge of the antenna location is usually not much of a problem, because the antenna is mounted on a mobile robot which can be assumed to have other sensors for self-localization purposes. Assuming knowledge of the precise RFID tag locations in the first place is much more of an issue. We therefore propose a novel method to simplify this calibration phase by getting rid of the dependence on known tag locations. For this, we propose an iterative calibration procedure that simultaneously estimates not only the sensor model but also the tag locations. Our experiments show that a sensor model learned in this way converges (up to a certain empirical error) to a sensor model learned with a known RFID tag map. Further, the accuracy of the RFID tag location estimates converge (up to a certain empirical error) to the true tag locations. We like to point out that this iterative procedure is only necessary if the sensor model should be learned without the reliance on a known RFID tag map. We quantitatively evaluate our approach and variants thereof in an office and a supermarket environment, also by comparing it to previously published sensor models. For this, we implemented and adapted a WiFi-based localization method, which constructs a 2D signal strength map by using Gaussian process regression. We additionally extend this approach by mapping the signal strength in pose space rather than in 2D. This is particular relevant for RFID-based localization, as the received signal strength can be quite sensitive to rotational changes of the antenna pose.

Given that the robot can localize objects, we move on to the question of how prior knowledge about usual object arrangements can be utilized by the robot to more efficiently carry out its tasks, particularly, when searching for objects in an unknown environment. As a motivation, consider the situation where you want to find a product in a supermarket where you have never been to before. Certainly, you will not just wander around randomly through the market nor will you systematically visit each so far unvisited aisle in the market until you find the product. Your search will rather be guided by the current observations and the expectations you have about how objects in supermarkets are usually arranged. You will have gained this knowledge by having seen quite a few markets throughout your life and noting certain strong spatial dependencies between certain groups of products. Thus, the main challenge that we need to address here is to, first, formalize this knowledge in a way such that it can be learned based on data of real supermarkets and, second, to devise a search strategy that leverages this knowledge in an appropriate way to speed up the search process. For this, we present two alternative search techniques. Our first approach is a reactive search technique that decides where to search next based on the objects in the robot's direct vicinity. The robot utilizes search heuristics in form of decision trees which classify the aisles at junctions into promising and non-promising directions and then continues to search in one of the promising directions. The decision trees are learned from data generated from optimal search paths in a training set of supermarkets. In contrast to the first approach, our second search technique is a more global, inference-based approach that takes into account all objects seen so far as well as the thus far discovered structure of the environment. Based on this, it first computes a distribution over the possible locations of the sought product. Then it selects a goal location by trading off the probability of finding the product at a certain location against the required distance to reach this location. It then continues its search by following the shortest path towards the selected goal location. Whenever new information becomes available, i.e., newly observed objects or

Figure 1.2: As a second contribution of this thesis, we present two approaches for efficiently finding objects in an unknown environment. Specifically, we investigate how background knowledge about usual object arrangements can be represented and utilized to more efficiently search for an object in an unknown environment. As an illustrative scenario we consider the search for a product in an unknown supermarket.

newly discovered parts of the environment, the robot will recompute this distribution and select a new goal location. The distribution is based on a model that, basically, takes into account how objects co-occur in different spatial contexts. The parameters of the model can be learned based on the layouts of supermarkets. While the technical details of these search techniques differ, they both are learned on the same training set of three supermarkets and then evaluated in a fourth test supermarket. For this, we collected data from four real supermarkets and modeled in detail the layout of the shelves and the placement of the products. The efficiency of both search techniques is quantitatively compared to a baseline approach that simply explores the environment until it finds the product, as well as to the performance of human subjects that searched in a real supermarket.

The inference-based search technique basically relies on co-occurrence statistics of objects in different spatial contexts. For example, it considers the probability that the sought object exists "in the same shelf" or "in a neighboring shelf" as other objects already observed while searching. While the parameters of this model are learned based on the layouts of supermarkets in a training set, the spatial relations utilized by this model are fixed and manually defined. Thus, in our final contribution of this thesis, we move on to a more ambitious learning scenario in which we wish to simultaneously learn both the spatial context and the co-occurrence of objects within the learned spatial contexts. That is, we wish to learn spatially coherent object constellations in complex multi-object scenes in an unsupervised way. As an application scenario we considered tabletop scenes and the object constellations are the covers consisting of, for example, a plate, a knife, and a mug. The identification of the relevant object constellations can be seen as parsing a scene or inferring the latent structure of a scene. However, for any given scene there exist multiple possible scene structures. For example, in the scene depicted in Fig. 1.3 we could argue that it simply contains twelve randomly placed objects. Alternatively, one might say that it contains just a single object constellation consisting of twelve objects. However, based on our prior knowledge of tabletop scenes, we would be tempted to say that it is much more likely that this scene contains three recurrent object constellations each consisting of four objects, namely a plate, a fork, a knife, and a mug. We will make this notion of "a more likely scene structure" precise, by defining a prior distribution over scene structures. This prior can then be used to evaluate the probability of each of the three above-mentioned alternative scene structures. Further, this prior can be updated by conditioning it on previously parsed scenes which results in a posterior predictive distribution over scenes. Thus, if the robot has never observed any such scene, it may consider the three alternative scene interpretations as more or less equally likely. However, if the robot has already parsed several scenes in which similar object constellations with four objects appear, then it might agree with our intuition that the scene most likely contains three object constellations with four objects each.

To be more specific, we define a novel hierarchical nonparametric Bayesian model for multi-object scenes. A basic building block of our model are so-called meta-objects which represent a distribution over object constellations. The actual object constellations on the table are assumed to have been sampled from such meta-objects. More specifically, meta-objects can be seen as probabilistic part-based models and they thus have an internal structure. Each part of a meta-object consists of (a) a spatial distribution that constrains the relative location of an object with respect to a meta-object reference frame, (b) a type distribution that constrains which objects should be placed there (e.g., a mug or plate), and (c) an activation probability that specifies the probability if an object should be placed at this location at all. A meta-object for the constellations shown in Fig. 1.3 would consist of four parts, e.g., a central part with a high probability for plates, a left part with a high probability for forks, etc. For sampling a constellation from this model, we first sample the

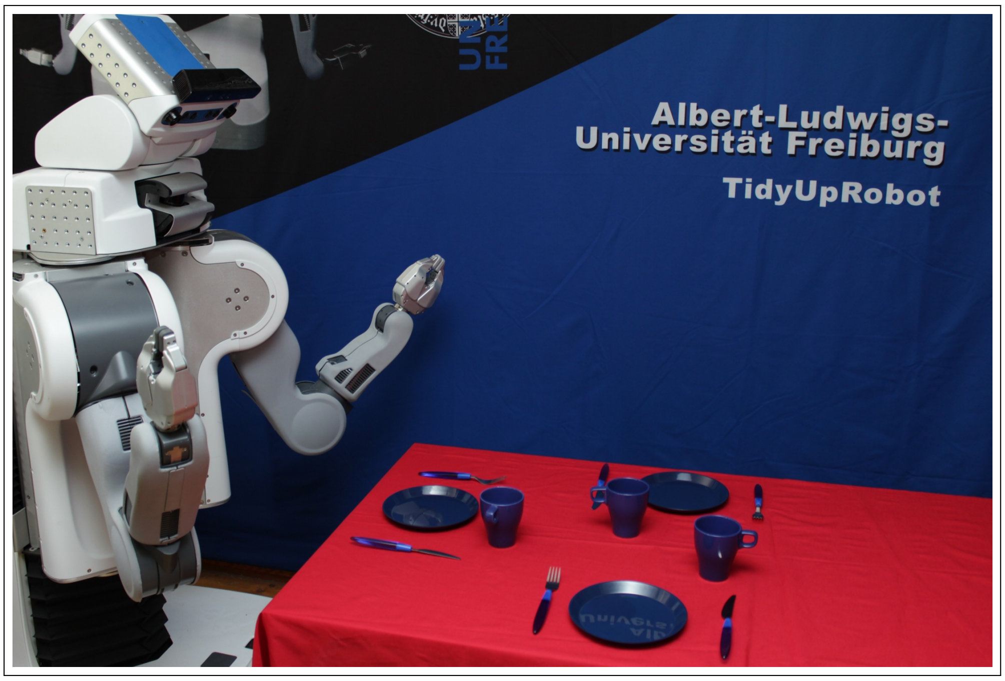


Figure 1.3: As a third contribution of this thesis, we investigate how a robot can learn spatially coherent object constellations in an unsupervised way by inferring the latent structure of everyday scenes. We consider tabletop scenes, in which the relevant object constellations correspond to the individual covers consisting of, for example, a plate, a knife, a fork, and a mug.

activation for each part. For each activated part, we then sample the relative location and the object type to be placed at this relative location, e.g., a mug or a plate. Hence, each part generates at most one object of a sampled object constellation. This object constellation is then transformed into the scene by sampling from a prior over transformations. We assume that a scene can contain multiple object constellations, such as the scene in the picture shown above, which contains three object constellations that have been transformed to three different places on the table. Further, we assume that there exist different *types* of object constellations, which might have different parametrizations of their part distributions. For example, one specific meta-object type might likely generate a constellation with a cereal bowl and a spoon, while another type would more likely generate object constellations like the ones mentioned above. Thus, parsing a scene corresponds to inverting this generative process by inferring the number of object constellations in a scene, their types, and their transformations or reference frames. Further, the objects on a table need to be associated to a certain meta-object instance as well as to a certain part of this meta-object instance. For example, the plate should be associated to the central part of its meta-object.

Please note, that we did not yet specify the *number* of meta-object *types*, the number of *parts* per meta-object, or the number of object *constellations* on the table. Actually, it would be desirable to avoid having to specify these numbers in advance and we would rather like to place a prior over these numbers and infer them from data. This is achieved by formulating our model as a nonparametric Bayesian model based on the Dirichlet process and the beta-Bernoulli process. By doing so, we do not even have to specify an upper bound for these numbers. Rather, we thereby assume that there exist infinitely many different meta-object types, each having infinitely many parts. In effect, the number of meta-object types and the number of meta-object parts are now subject to inference as well and we can set certain hyper-parameters which express our prior belief with respect to these numbers. This allows for a probabilistic treatment of the effective model complexity which thereby can adapt to the data complexity. This also has practical benefits in the context of lifelong learning for service robots. As the number of object constellations is not fixed in our model, the robot is able to recognize and integrate previously unseen object constellations into its model in an open-ended fashion and within a single coherent probabilistic framework.

1.1 Contributions

The work presented in this thesis contributes to mainly three areas in the context of mobile robotics: (a) RFID-based localization and mapping, (b) object search in unknown environments, and (c) scene analysis. In particular, we like to highlight the following contributions.

1.1.1 RFID-based Localization and Mapping

- A novel probabilistic sensor model for RFID-based localization and mapping. In contrast to previously published models, our model considers both signal strength and tag detection events.

- A novel unsupervised procedure for simultaneously learning both the sensor model and the tag locations in an iterative manner. This generalizes a supervised non-iterative method for learning the sensor model which is only applicable when the tag locations are known beforehand.

- For evaluation purposes, an adaption of a previously published sensor-centric RFID sensor model.

- For evaluation purposes, an adaption of a previously published WiFi-based sensor model based on Gaussian process regression. The model is further extended to map the signal strength in pose space rather than in 2D.
- An extensive evaluation of our method in an office and a supermarket environment.

1.1.2 Object Search in Unknown Environments

- A novel reactive search technique that utilizes search heuristics in form of decision trees which classify the aisles at junctions into promising and non-promising directions. The robot then continues to search in one of the promising directions.
- A technique to learn these decision trees based on data generated from optimal search paths in training environments.
- An inference-based search technique that maintains a distribution over the possible location of the object. This distribution takes into account all objects seen so far and the thus far discovered structure of the environment.
- A technique to learn the parameters of this model based on the layout of training environments.
- An extensive evaluation of both techniques, including a comparison to a baseline technique, that explores the supermarket until it finds the product, and to the performance of human subjects that participated in a field study conducted in a real market.

1.1.3 Scene Analysis

- A novel probabilistic approach to reason about the latent structure of multi-object scenes in terms of object constellations. For this, we propose a novel hierarchical nonparametric Bayesian model based on nested and hierarchical variants of the Dirichlet process (or Chinese restaurant process) and the beta-Bernoulli process (or Indian buffet process).
- A Markov chain Monte Carlo procedure to sample from the posterior distribution over the latent scene structures when conditioned on the observed objects of the scenes. This includes efficient MCMC moves that propose to change the correspondence variables of several observed objects at a time. Further, we introduce a novel

top-down bottom-up proposal for the MCMC sampler that takes advantage of both the observed data and the currently instantiated representations in the model.

- An evaluation of the proposed MCMC technique for obtaining samples from the posterior distribution of our model when conditioned on tabletop scenes from both synthetic and real-world data sets.

1.2 Publications

Parts of the work presented in this thesis have been previously published in the following journal and conference publications. All publications were peer-reviewed except where noted. For each publication, we point out the related chapter in this thesis.

Journal

- **Dominik Joho**, Martin Senk, and Wolfram Burgard. Learning search heuristics for finding objects in structured environments. *Robotics and Autonomous Systems (RAS)*, 59(5):319–328, May 2011.
  Related: Chap. 4

Conference

- **Dominik Joho**, Gian Diego Tipaldi, Nikolas Engelhard, Cyrill Stachniss, and Wolfram Burgard. Nonparametric Bayesian models for unsupervised scene analysis and reconstruction. In *Proc. of Robotics: Science and Systems (RSS)*, Sydney, Australia, 2012.
  Related: Chap. 5

- **Dominik Joho** and Wolfram Burgard. Searching for objects: Combining multiple cues to object locations using a maximum entropy model. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 723–728, Anchorage, AK, USA, May 2010.
  Related: Chap. 4

- **Dominik Joho**, Christian Plagemann, and Wolfram Burgard. Modeling RFID signal strength and tag detection for localization and mapping. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 3160–3165, Kobe, Japan, May

2009.
Related: Chap. 3

- **Dominik Joho**, Martin Senk, and Wolfram Burgard. Learning wayfinding heuristics based on local information of object maps. In *Proc. of the Europ. Conf. on Mobile Robots (ECMR)*, pages 117–122, Mlini/Dubrovnik, Croatia, September 2009.
Related: Chap. 4

Workshop

- **Dominik Joho**, Gian Diego Tipaldi, Nikolas Engelhard, Cyrill Stachniss, and Wolfram Burgard. Unsupervised scene analysis and reconstruction using nonparametric Bayesian models. In *Proc. of the Workshop on Robots in Clutter at Robotics: Science and Systems (RSS)*, Sydney, Australia, 2012.
Related: Chap. 5

Extended Abstract (not peer-reviewed)

- **Dominik Joho**, Gian Diego Tipaldi, Nikolas Engelhard, Cyrill Stachniss, and Wolfram Burgard. Unsupervised scene analysis using semiparametric Bayesian models. In *Extended Abstracts of Spatial Cognition (SC)*, Kloster Seeon, Germany, 2012.
Related: Chap. 5

1.3 Collaborations

The search strategy based on decision trees is one of two search strategies presented in Chap. 4 and was originally addressed in the co-supervised Bachelor's thesis of Martin Senk [63]. The same chapter presents results of a field study conducted with human participants in a real supermarket. This field study was carried out by Christopher Kalff and colleagues of the Center for Cognitive Science at the University of Freiburg. To track the participants in the supermarket, Christopher Kalff and colleagues used our proposed RFID-based localization system. The resulting trajectories and performance metrics were then used by us for a comparison with our proposed search technique. In Chap. 5, the segmentation and classification of the pointclouds of the tabletop scenes has been implemented by Nikolas Engelhard.

1.4 Notation

In Tab. 1.1 we illustrate notational conventions used throughout this thesis. Please note that there is an ambiguity whether a boldface $\mathbf{x}$ denotes (a) a vector, (b) the outcome $x_1, \dots, x_n$ of a collection of scalar random variables, or (c) the outcome $\mathbf{x}_1, \dots, \mathbf{x}_n$ of a collection of vector-valued random variables. It should be clear from the context which case applies and we will therefore not further resolve this issue.

Table 1.1: Some notational conventions used in this thesis.

Symbol	**Meaning**
	Algebra and Geometry
x	scalar
$\mathbf{x}$	vector
$\mathbf{x}^{\mathsf{T}}$	transposed vector
$\mathbf{X}$	matrix
$a \propto b$	a is proportional to b
	Probability Theory and Monte Carlo Methods
X	random variable
$\mathbf{X}$	multiple random variables
x	possible outcome of a random variable, e.g. $X = x$
$\mathbf{x}$	multiple outcomes of the corresponding random variables, e.g. $\mathbf{X} = \mathbf{x}$
$\mathbf{x}_{1:t}$	sequence of outcomes, $\mathbf{x}_{1:t} = x_1, \dots, x_t$
$p(x)$	probability mass function or probability density function
$p(z \mid x)$	conditional probability of z given x
$x \sim \mathrm{Dist}$	x is distributed according to Dist
$\delta_x(\cdot)$	Dirac delta function centered at x
$\mathbb{E}_p(X)$	expectation (expected value) of random variable X wrt. the distribution p
$\mathbb{H}(p)$	entropy of the distribution p
	Probability Distributions and Stochastic Processes
BP	beta process
Bern	Bernoulli distribution
BeP	Bernoulli process
CRP	Chinese restaurant process
Dir	Dirichlet distribution
DP	Dirichlet process
GP	Gaussian process
IBP	Indian buffet process
Mult	multinomial distribution
$\mathcal{N}$	normal distribution (Gaussian distribution)
$\mathcal{NW}$	normal-Wishart distribution
Pois	Poisson distribution

CHAPTER 2

Basics

We review some of the theoretical foundations underlying the techniques presented in later chapters. We aim at giving a concise yet readable introduction to the various topics and we provide pointers to the literature for the interested reader who desires a more in-depth treatment of these topics.

We start by reviewing the basics of probability theory, which is the foundation for many of the topics to follow. We then proceed with an introduction to Monte Carlo methods, including particle filtering and Markov chain Monte Carlo (MCMC) methods. Particle filtering is a frequently used technique for mobile robot localization. It will be utilized in Chap. 3 for estimating the trajectory of a sensor platform moving through an RFID-enabled environment. MCMC techniques will be of importance in Chap. 5 for performing posterior inference in a generative probabilistic model of tabletop scenes. We then describe decision trees, that will be used as a way to learn and represent search heuristics in Chap. 4. Next, we discuss maximum entropy models, which will be used in the same chapter for modeling a distribution over the possible locations of an object the robot is searching for. Finally, we have a section on Bayesian nonparametrics, which covers three commonly used stochastic processes. The first two processes, the Dirichlet process and the beta-Bernoulli process, are the corner stones for our proposed hierarchical model of tabletop scenes that we will describe in Chap. 5. The third stochastic process, the Gaussian process, is used in Chap. 3 to represent signal strength maps as an alternative RFID localization method that we evaluate alongside our proposed antenna-centric approach.

2.1 Basics of Probability Theory

Mobile robots are autonomous systems that act in a flexible and adaptive way by observing the world and by making their actions dependent on the current state of the world. For this, the robot needs to acquire information about the current state of the world, including its own state, and it needs to reason about how the state might evolve over time as well as how its own actions might influence this state. Obviously, all three stages – sensing, planning, acting – involve substantial uncertainties. A robot never directly observes the true state of the world. Rather, it has to infer this state from sensor data, which might be inherently ambiguous with regard to this state and, additionally, it might be subject to measurement noise. Planning needs to deal with uncertainties, as the robot can never actually predict the future states of the world with absolute certainty. Likewise, the outcome of its own actions are uncertain, too, be it because a wheel slips on the ground or its gripper misses the bottle on the table. A formal system to deal with uncertainty in a consistent way is probability theory. Not surprisingly, there exists a wide range of techniques in mobile robotics that have their theoretical foundation in probability theory [74] and many of the techniques being used in this thesis are also based on probability theory. In the following, we will therefore shortly review the basics of probability theory [59, 74].

Like propositional logic, probability theory deals with statements. However, in contrast to propositional logic, these statements are not considered to be either true or false but to be more or less likely. Hence, a statement is assigned a continuous value, a probability, in the interval $[0, 1]$. Further, probability theory formalizes a consistent way for calculating, or inferring, probabilities for statements based on the probabilities of other statements. Equivalently, one might say probability theory deals with random events and the statements are about the possible outcome of events. For example, consider the cast of a die. Then X could be the random event that denotes the number of dots the die will show after it has been cast. Then $X = 2$ is a statement about the outcome of this event, namely that the die will show 2 dots. If we think it is a fair die, we might assign the probability $p(X = x) = \frac{1}{6}$ for all possible outcomes $x \in \{1, \dots, 6\}$. In this example, X is a discrete random variable with sample space $\mathcal{X} = \{1, \dots, 6\}$ and with a uniform discrete distribution p.

More formally, a random variable X is defined over a sample space $\mathcal{X}$ that defines all possible outcomes and p is a probability distribution over the sample space $\mathcal{X}$. For discrete random variables, p is a probability mass function (PMF) and we denote by $p(X = x)$ the probability that the random variable will assume the value x. Often, we will simply write $p(x)$ for $p(X = x)$. For continuous variables, p is a probability density function (PDF) and we denote by $p(y)$ the value of the density function at y and by $p(Y \in A)$ we denote the probability of the event that the random variable Y assumes a value in the subset $A \subseteq \mathcal{Y}$ with $\mathcal{Y}$ being the corresponding sample space. Probability distributions need to satisfy certain consistency properties. First, a probability is a real in the interval $[0, 1]$, hence

$$p(X = x) \in [0,1] \qquad \text{for all } x \in \mathcal{X} \text{ and } p \text{ is a PMF} \tag{2.1}$$

$$p(Y \in A) \in [0,1] \qquad \text{for all } A \subseteq \mathcal{Y} \text{ and } p \text{ is a PDF} \tag{2.2}$$

Next, the probabilities for discrete random variables sum to one and the density function for continuous variables integrates to one.

$$\sum_{x \in \mathcal{X}} p(X = x) = 1 \qquad p \text{ is a PMF} \tag{2.3}$$

$$\int_{y \in \mathcal{Y}} p(y)\, dy = 1 \qquad p \text{ is a PDF} \tag{2.4}$$

Further, probabilities are additive in the sense that for mutually exclusive events $A = \{a_1, \ldots, a_n\}$ with $a_i \subseteq \mathcal{X}$ and $a_i \cap a_j = \emptyset$ for $i \neq j$ we have for a PMF p

$$p(X = A) = p(X = a_1 \text{ or } \ldots \text{ or } X = a_n) \tag{2.5}$$

$$= p(X = a_1) + \cdots + p(X = a_n) \tag{2.6}$$

and likewise for a PDF p with the corresponding adjustments in notation, e.g. $p(Y \in A)$ instead of $p(X = A)$ and the $\{a_1, \ldots, a_n\}$ then form a partition of the subset $A \subseteq \mathcal{Y}$. For ease of exposition, we assume for the rest of this section that all variables are discrete. Some further properties that follow from the above-said are the certain event

$$p(\mathcal{X}) = 1 \tag{2.7}$$

the impossible event

$$p(\emptyset) = 0 \tag{2.8}$$

and the complementary event

$$p(A) = 1 - p(\mathcal{X} \backslash A). \tag{2.9}$$

In any practical setting, we will have to deal with multiple random variables, e.g. X and Y, and their joint distribution $p(x, y)$ which corresponds to the event that $X = x$ holds *and* that $Y = y$ holds. A joint distribution can be factorized into conditional distributions according to the chain rule

$$p(x, y) = p(x \mid y)p(y) = p(y \mid x)p(x). \tag{2.10}$$

Here, the conditional distribution $p(x \mid y)$ is the distribution for the random variable X given that we know (with certainty) that $Y = y$ is the case. If the knowledge of the value of Y tells us nothing about the possible values of X, then this would keep the distribution for X unchanged such that $p(x \mid y) = p(x)$ and we say these variables are independent. Hence in this case we would have

$$p(x, y) = p(x)p(y) \qquad\qquad X \text{ and } Y \text{ are independent.} \tag{2.11}$$

An important variant thereof is conditional independence, in which two variables X and Y are dependent, but become independent when given knowledge of the value of a third random variable Z. In this case, we say X and Y are conditionally independent given Z and we have

$$\begin{aligned} p(x, y, z) &= p(x, y \mid z)p(z) && (2.12) \\ &= p(x \mid z)p(y \mid z)p(z) \qquad X \text{ and } Y \text{ are cond. indep. given } Z. && (2.13) \end{aligned}$$

In practice, we will often deal with distributions with more than two variables. One way of modeling complex joint distributions over multiple variables are Bayesian networks in which we assume certain conditional independencies between these variables and factorize the joint distribution into conditional distributions. These conditional distributions can then be modeled by standard distributions such as the Gaussian distribution or the Poisson distribution. The joint distribution can be represented as a directed acyclic graph like the one depicted in Fig. 2.1a. The joint factorizes into conditional distributions for each of the nodes, where the random variable of a node is conditioned on the random variables of its predecessor nodes or parent nodes. Hence, for the example in Fig. 2.1a we have

$$p(a, b, c, d) = p(d \mid b, c)p(b \mid a)p(c \mid a)p(a). \tag{2.14}$$

A further important concept is Bayes' rule, which directly follows from the definition

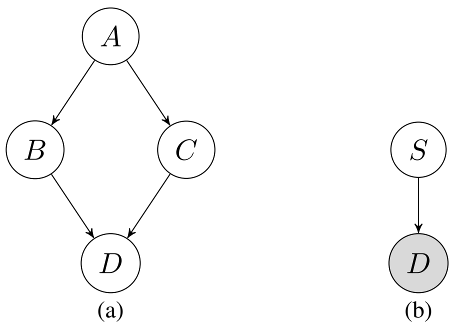


Figure 2.1: Bayesian networks can be represented as directed acyclic graphs in which nodes represent random variables and edges encode dependencies. Gray nodes correspond to observed variables and white nodes to latent variables.

of conditional probabilities. Consider the following situation in which we have a random variable D, which represents observable data that the robot may have obtained from sensor measurements. Further, we have a latent random variable S, which represents some hypothesis about the state of the world, that the robot likes to infer in the light of the data. That is, we like to compute the conditional probability $p(s \mid d)$ of a state given the data. If the joint distribution is modeled by a Bayesian network such as the one depicted in Fig. 2.1b, then the joint distribution factorizes as $p(d, s) = p(d \mid s)p(s)$. This can be interpreted as a causal model in which the state of the world influences the sensor data. We now need to "invert" this causal relationship, by reasoning about the possible state of the world when given the data. For this, we can apply Bayes' rule

$$p(s \mid d) = \frac{p(d \mid s)p(s)}{p(d)} = \frac{p(d \mid s)p(s)}{\sum_{s'} p(d \mid s')p(s')}. \tag{2.15}$$

Here, the equivalence $\sum_{s'} p(d \mid s')p(s') = p(d)$ is an example for the law of total probability in which we integrate out a variable. Bayes' rule illustrates the general principle for posterior inference in Bayesian network with multiple variables. We have a joint distribution $p(\mathbf{x}, \mathbf{z})$ over several observed variables $\mathbf{z} = \{z_1, \ldots, z_n\}$ and several latent variables $\mathbf{x} = \{x_1, \ldots, x_m\}$ and we like to infer the distribution over the latent variables given the observed variables based on the definition of the joint distribution

$$p(\mathbf{x} \mid \mathbf{z}) = \frac{p(\mathbf{z}, \mathbf{x})}{p(\mathbf{z})} = \frac{p(\mathbf{z}, \mathbf{x})}{\sum_{x'_1} \cdots \sum_{x'_m} p(\mathbf{z}, x'_1, \ldots, x'_m)}. \tag{2.16}$$

In practice, though, it might be infeasible to compute the sum in the denominator. However, any quantity that depends on the distribution $p(\mathbf{x} \mid \mathbf{z})$ can be approximated by drawing samples from this distribution and basing the calculations on the obtained samples. This is the basic idea of Monte Carlo methods, which we will describe in the following.

2.2 Monte Carlo Methods

Monte Carlo methods [2, 46, 74] can be used to approximate intractable integrals or sums, such as expectations with respect to a probability distribution $p(x)$. In this case, the main idea is to obtain a set of samples $\{x^{(1)}, \ldots, x^{(N)}\}$ from a distribution $p(x)$ over the sample space $\mathcal{X}$ and then to approximate the expectation

$$\mathbb{E}_p(f(x)) = \int_{x \in \mathcal{X}} f(x)p(x)\,dx \tag{2.17}$$

by the Monte Carlo approximation based on the samples

$$\mathbb{E}_N(f(x)) = \frac{1}{N}\sum_{i=1}^{N} f(x^{(i)}). \tag{2.18}$$

The main problem here is to obtain samples from the distribution $p(x)$. If the distribution $p(x)$ is of some standard form, such as a Gaussian, then there might exist dedicated algorithms for directly drawing samples. However, for the general case of arbitrarily shaped distributions or density functions, we need more generally applicable methods for drawing samples. In the following, we will describe two widely used techniques for this: importance sampling and Markov chain Monte Carlo. Further, we describe a sequential Monte Carlo method for sampling from a dynamic Bayesian network that forms the basis for mobile robot localization.

2.2.1 Importance Sampling

In importance sampling [14, 74], we side-step the impracticality of directly sampling from $p(x)$ by introducing an auxiliary proposal distribution $q(x)$ from which we can easily sample. The samples from the proposal $q(x)$ are then carefully adjusted such that they appear to be sampled from the target distribution $p(x)$ that we are actually interested in. For this, the proposal distribution $q(x)$ must have the same support $\mathcal{X}$ as the target distribution. The following derivation is based on [14]. Remember that our initial attempt was to compute the integral of Eq. (2.17), which we could rewrite by introducing the proposal distribution

$q(x)$ as follows

$$\mathbb{E}_p(f(x)) = \int f(x)p(x)\,dx \tag{2.19}$$
$$= \int f(x)\frac{p(x)}{q(x)}q(x)\,dx. \tag{2.20}$$

Let us define the so-called importance weight as $w(x) \equiv \frac{p(x)}{q(x)}$ and remember that

$$\int w(x)q(x)\,dx = \int \frac{p(x)}{q(x)}q(x)\,dx = 1. \tag{2.21}$$

Then we have

$$\mathbb{E}_p(f(x)) = \int f(x)\frac{p(x)}{q(x)}q(x)\,dx \tag{2.22}$$
$$= \int f(x)w(x)q(x)\,dx \tag{2.23}$$
$$= \frac{\int f(x)w(x)q(x)\,dx}{\int w(x)q(x)\,dx} \tag{2.24}$$

Both integrals appearing in Eq. (2.24) can be approximated by a Monte Carlo estimate with respect to N samples $\{x^{(1)}, \ldots, x^{(N)}\}$ drawn from $q(x)$, which leads to

$$\mathbb{E}_N(f(x)) = \frac{\frac{1}{N}\sum_i f(x^{(i)})w(x^{(i)})}{\frac{1}{N}\sum_i w(x^{(i)})} \tag{2.25}$$
$$= \sum_i f(x^{(i)})\frac{w(x^{(i)})}{\sum_{i'} w(x^{(i')})} \tag{2.26}$$
$$= \sum_i f(x^{(i)})\tilde{w}(x^{(i)}), \tag{2.27}$$

where

$$\tilde{w}(x^{(i)}) \equiv \frac{w(x^{(i)})}{\sum_{i'} w(x^{(i')})} \tag{2.28}$$

is the normalized importance weight of sample $x^{(i)}$. Please note, that this method is also applicable if the target distribution $p(x)$ is known only up to a normalizing constant, because this constant would cancel in Eq. (2.24) where it appears in each occurrence of the importance weight $w(x)$.

To summarize, we draw samples from $q(x)$ and compute the normalized weights for each sample based on the proposal density and the target density. Intuitively, we may say that the target distribution is now (approximately) represented by the set of weighted samples. Especially, the samples can be used to compute Monte Carlo estimates of expectations with respect to the original target distribution.

However, there is the drawback that importance sampling depends on a well-designed proposal distribution that should closely match the target distribution over the whole sample space. Otherwise, we would "loose" many samples in low-probability regions of the target distributions. For arbitrarily shaped target distributions it may become difficult to find appropriate proposal distributions from which we can easily sample. Further, this will become even more problematic for high-dimensional target distributions. In such situations, we can apply Markov chain Monte Carlo methods, that we describe in the next section.

2.2.2 Markov Chain Monte Carlo

The main difficulty of importance sampling was to find a proposal distributions that *globally* matches the target distribution as close as possible. Further, the samples have been drawn independently of each other. If one sample happens to be in a high-probability region of the target distribution, then this will have no influence on subsequent samples. Roughly speaking, we would like to utilize this information and try to "explore" or search the space in this region. This idea is taken up by Markov chain Monte Carlo (MCMC) methods, in which we can utilize proposal distributions $q(x \mid x^{(i)})$ that may depend on the latest sample $x^{(i)}$. For example, we could sample from a proposal distribution that puts most of its probability mass around a local neighborhood of $x^{(i)}$. We thereby get a sequence of samples that move through the state space in small steps. However, if these samples should represent draws from the target distribution, we must take care to also make this sequence dependent on the target distribution in an appropriate way.

One commonly used MCMC method is the Metropolis-Hastings algorithm [2]. We are given a target distribution $p(x)$ known up to a normalization constant, a proposal distribution $q(x' \mid x)$, and an arbitrary initial sample $x^{(0)}$. We then iterate the following. Based on the current sample $x^{(i)}$, we propose a new sample $x^\star$ drawn from the proposal $q(x^\star \mid x^{(i)})$.

We then compute the acceptance ratio

$$R = \frac{p(x^\star)q(x^{(i)} \mid x^\star)}{p(x^{(i)})q(x^\star \mid x^{(i)})}, \tag{2.29}$$

and set the new sample $x^{(i+1)}$ to

$$x^{(i+1)} = \begin{cases} x^\star & \text{with probability } \min(1, R) \\ x^{(i)} & \text{with probability } 1 - \min(1, R) \end{cases}. \tag{2.30}$$

Thus, the proposed sample $x^\star$ is either accepted or rejected with a probability depending on the acceptance ratio R. If the sample is accepted, the Markov chain “moves” to this proposed location, otherwise, it “stays” at the current location. The obtained sample is then added to the set of samples that represent draws from the target distribution. Thus, if the chain stays at the current location, the obtained sample is once again added to the set of samples.

Another MCMC method is Gibbs sampling, which can be derived as a special case of the Metropolis-Hastings algorithm. The following derivation is based on [2]. In Gibbs sampling, we have a multidimensional state space $\mathbf{x} = (x_1, \ldots, x_n)$ and we propose a new sample by changing only a single variable x_j while the other variables, denoted as $\mathbf{x}_{[-j]}$, remain unchanged. Thus, based on the i-th sample $\mathbf{x}^{(i)}$ we would propose a sample $\mathbf{x}^\star = (x_1^{(i)}, \ldots, x_{j-1}^{(i)}, x_j^\star, x_{j+1}^{(i)}, \ldots, x_n^{(i)})$. The proposal distribution draws $x_j^\star$ from the full conditional distribution $p(x_j^\star \mid \mathbf{x}_{[-j]}^{(i)})$ of the *target* distribution, and hence

$$q_j(\mathbf{x}^\star \mid \mathbf{x}^{(i)}) = \begin{cases} p(x_j^\star \mid \mathbf{x}_{[-j]}^{(i)}) & \text{if } \mathbf{x}_{[-j]}^\star = \mathbf{x}_{[-j]}^{(i)} \\ 0 & \text{otherwise} \end{cases}. \tag{2.31}$$

The acceptance ratio for a sample proposed in this way is

$$\begin{aligned} R &= \frac{p(\mathbf{x}^\star)}{p(\mathbf{x}^{(i)})} \frac{q_j(\mathbf{x}^{(i)} \mid \mathbf{x}^\star)}{q_j(\mathbf{x}^\star \mid \mathbf{x}^{(i)})} && (2.32) \\ &= \frac{p(\mathbf{x}^\star)}{p(\mathbf{x}^{(i)})} \frac{p(x_j^{(i)} \mid \mathbf{x}_{[-j]}^\star)}{p(x_j^\star \mid \mathbf{x}_{[-j]}^{(i)})} && (2.33) \\ &= \frac{p(x_j^\star \mid \mathbf{x}_{[-j]}^{(i)})p(\mathbf{x}_{[-j]}^{(i)})}{p(x_j^{(i)} \mid \mathbf{x}_{[-j]}^\star)p(\mathbf{x}_{[-j]}^\star)} \frac{p(x_j^{(i)} \mid \mathbf{x}_{[-j]}^\star)}{p(x_j^\star \mid \mathbf{x}_{[-j]}^{(i)})} && (2.34) \end{aligned}$$

$$= \frac{p(\mathbf{x}^{(i)}_{[-j]})}{p(\mathbf{x}^{\star}_{[-j]})} \tag{2.35}$$

$$= 1. \qquad \text{because } \mathbf{x}^{(i)}_{[-j]} = \mathbf{x}^{\star}_{[-j]} \tag{2.36}$$

Thus, the proposed sample is always accepted. During Gibbs sampling we would apply this step for each dimension or variable of the state space, either in a fixed or random order. The main drawbacks of a Gibbs sampler are twofold. First, we must be able to draw samples from the full conditional distribution of the target distribution. If it is impossible to directly sample from this conditional, then this could be side-stepped by using Metropolis-Hastings steps to sample from the conditional. This is known as “Metropolis within Gibbs”. Second, because we are only changing a single variable at a time the sampler might easily get stuck in local minima if there exist strong dependencies between several variables. In this case, it is beneficial to extend the principle of Gibbs sampling to the case where we jointly propose new values for these strongly correlated variables. This is known as blocked Gibbs sampling. However, as more variables are clumped together in one block, it might become increasingly difficult to devise a sampling strategy to jointly draw new values for these variables from the implied conditional distribution of the target distribution.

In practical applications of MCMC methods, one usually ignores the first couple of samples during a “burn-in” period. This is, because the initial sample $x^{(0)}$ usually lies in a region of low probability and the Markov chain first has to reach regions of high probability and, therefore, the first samples are not representative for the target distribution. Further, one could take into account only every n-th sample of the chain to minimize the correlation between the samples. In contrast to importance sampling, MCMC methods can take advantage of having found regions of high probability by exploring these regions with local moves from the proposal. However, if the target distribution is multimodal and the modes are far apart, it may easily happen that the chain only explores a single mode. One could try to avoid this by starting several independent chains from different initial locations. Further, one could propose more global moves from time to time in the hope of reaching an area close to another mode. Hence, it is apparent that also for MCMC methods it can be difficult to come up with well-designed proposal distributions.

2.2.3 Monte Carlo Localization for Mobile Robots

Monte Carlo localization [21, 74] is an approach for tracking the pose of a robot by formalizing the tracking problem as in inference problem in a dynamic Bayesian network such as

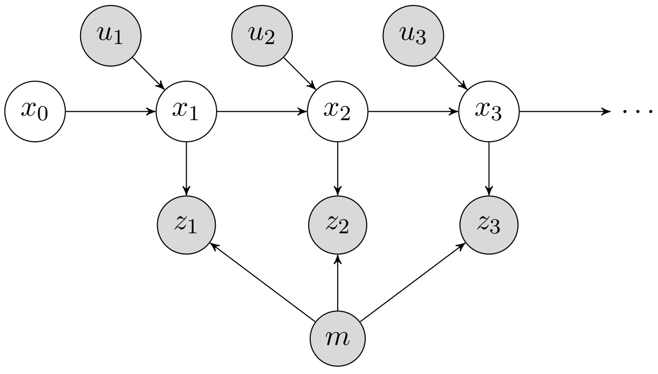


Figure 2.2: The Bayesian network used for mobile robot localization. Gray nodes denote observed variables.

the one depicted in Fig. 2.2. The robot's pose is modeled as a latent variable x_t that evolves over time in discrete steps. The motion model $p(x_t \mid x_{t-1}, u_t)$ describes how the pose at a certain time t depends on the previous pose x_{t-1} and the latest motion command or control u_t of the robot. Further, this distribution could depend on the map m to model impossible transitions through walls, though for simplicity we omit this here. At each time step, the robot takes a sensor measurement z_t and the senor model $p(z_t \mid x_t, m)$ describes how a measurement depends on the current pose of the robot and a given map m of the environment. For localization purposes, we are interested in the belief $\mathrm{bel}(x_t)$ as a distribution over the current location of the robot. This corresponds to the posterior distribution

$$\mathrm{bel}(x_t) = p(x_t \mid z_{1:t}, u_{1:t}, m) \tag{2.37}$$

over the current pose of the robot conditioned on the map and all past measurements and controls. Further, all past states have been integrated out. To compute the belief for the next time step, we first compute the predictive distribution $\overline{\mathrm{bel}}(x_{t+1})$ which is the distribution that takes into account the next control u_{t+1} but does not yet include the information about the next measurement

$$\overline{\mathrm{bel}}(x_{t+1}) = p(x_{t+1} \mid z_{1:t}, u_{1:t+1}, m) \tag{2.38}$$

$$= \int_{x_t} p(x_{t+1} \mid x_t, u_{t+1}) \, \mathrm{bel}(x_t) \, dx_t. \tag{2.39}$$

Finally, the predicted belief is adjusted by incorporating the information about the new measurement z_t and hence

$$\mathrm{bel}(x_{t+1}) \propto p(z_{t+1} \mid x_{t+1})\overline{\mathrm{bel}}(x_{t+1}). \tag{2.40}$$

In Monte Carlo localization the state distribution is represented nonparametrically by a set of weighted samples called particles and the resulting method is also called a particle filter. As argued in [74], the particle filter can actually be thought of estimating a distribution $\mathrm{bel}(x_{1:t})$ over the full trajectory $x_{1:t} = x_1, \ldots, x_t$ instead of just the filtering distribution $\mathrm{bel}(x_t)$ over the last pose x_t. Thus, the set of weighted particles $\{x_{1:t}^{(i)}, w_t^{(i)}\}_{i=1}^N$ represents an estimate of $\mathrm{bel}(x_{1:t})$. However, if we simply ignore the past states $x_{1:t-1}^{(i)}$ of each particle, then $\{x_t^{(i)}, w_t^{(i)}\}_{i=1}^N$ also defines an estimate of the desired filtering distribution $\mathrm{bel}(x_t)$ over the current state.

The particle filter then starts with an initial distribution over the first time step and then adjusts the particle set to represent a distribution over successively longer histories. In detail, this works as follows:

- *Initialization:* Initialize the particle set once by drawing each particle from the prior $x_0^{(i)} \sim p(x_0)$ and set the weights uniformly to $w_0^{(i)} = \frac{1}{N}$.
- *Prediction:* For each particle $x_{1:t}^{(i)}$, sample a new state for the next time step by sampling from the motion model $x_{t+1}^{(i)} \sim p(x_{t+1} \mid x_t^{(i)}, u_t)$ and append this state to the history of the particle.
- *Update:* Compute the new importance weight of each particle $w_{t+1}^{(i)} = p(z_{t+1} \mid x_{t+1}^{(i)}, m)$ and then normalize the weights.
- *Resampling:* Resample N particles from the particle set according to their current weights. Then set $w_{t+1}^{(i)} = \frac{1}{N}$ for each particle.

The weighing of the particles in the update step accounts for the fact that the particles have been sampled from the predictive distribution

$$\overline{\mathrm{bel}}(x_{1:t+1}) = p(x_{t+1} \mid x_t, u_{t+1})\, \mathrm{bel}(x_{1:t}) \tag{2.41}$$

but we are actually interested in obtaining samples from the updated belief

$$\mathrm{bel}(x_{1:t+1}) = \eta\, p(z_{t+1} \mid x_{t+1})\overline{\mathrm{bel}}(x_{1:t+1}), \tag{2.42}$$

that also takes into account the latest sensor measurement. We can use the principle of importance sampling to correct for this mismatch by computing an importance weight for the particles by dividing the target distribution by the proposal distribution

$$\begin{aligned} w_{t+1}^{(i)} &= \frac{\operatorname{bel}(x_{1:t+1}^{(i)})}{\overline{\operatorname{bel}}(x_{1:t+1}^{(i)})} && (2.43) \\ &= \frac{\eta\, p(z_{t+1} \mid x_{t+1}^{(i)}, m)\overline{\operatorname{bel}}(x_{1:t+1}^{(i)})}{\overline{\operatorname{bel}}(x_{1:t+1}^{(i)})} && (2.44) \\ &= \eta\, p(z_{t+1} \mid x_{t+1}^{(i)}, m). && (2.45) \end{aligned}$$

Please note, that the normalizer η is of no importance, as it cancels during the normalization of the weights.

The resampling step transforms the non-uniformly weighted particles into a set of uniformly weighted particles. The set may now contain duplicates of previously highly weighted particles and particles with a low importance weight may get lost. This is a crucial step, because we only have a finite set of particles and, therefore, it is necessary to concentrate the particles in regions of high probability.

In an important variant of the particle filter the resampling step is delayed. In this case, the importance weight is adjusted to take into account the previous weight such that $w_{t+1}^{(i)} = p(z_{t+1} \mid x_{t+1}^{(i)}, m)\, w_t^{(i)}$. The weights can still be normalized in each iteration. A common criterion to decide when to perform a resampling step is the number of effective particles defined as

$$N_{\text{eff}} = \frac{1}{\sum_i \left(w_t^{(i)}\right)^2}. \qquad (2.46)$$

If this number falls below a certain threshold, typically $\frac{N}{2}$, a resampling step is performed and the weights are again initialized uniformly as $\frac{1}{N}$.

2.3 Decision Tree Learning

Decision trees [52, 8, 59] can be used for supervised classification, in which we are given a set of labeled training examples $\{(c^{(i)}, a_1^{(i)}, \ldots, a_n^{(i)})\}_{i=1}^m$. Here, we assume that the labels $c^{(i)} \in \{0, 1\}$ are binary and their values denote if the corresponding example is a positive or negative one. Further, we have a set of attributes $\mathcal{A} = \{A_1, \ldots, A_n\}$, where each attribute defines a finite set of possible values $A_i = \{v_{i,1}, \ldots, v_{i,k}\}$. Besides the label, each training example is additionally described by a set of attribute values $a_i \in A_i$.

A decision tree can be used to classify a new example or observation based on its attribute values. For this, the observation is propagated top-down through the tree. Each node of the tree corresponds to a test for a certain attribute A_i of the observation. Depending on the attribute value a_i of the observation it is propagated to one of the child nodes. The leaf nodes are associated with a class label that defines the classifier output.

To learn the decision tree, it is constructed top-down and each node is associated with a set of positive E_p and negative E_n examples of the training set and a set $\mathcal{A}_u$ of yet untested attributes. At each node, an attribute $A \in \mathcal{A}_u$ is chosen that maximizes the information gain

$$G(A) = I\left(|E_p|, |E_n|\right) - \sum_{v \in A} \frac{\left|E_{p(v)}\right| + \left|E_{n(v)}\right|}{|E_p| + |E_n|} I\left(\left|E_{p(v)}\right|, \left|E_{n(v)}\right|\right) \tag{2.47}$$

where

$$I(p, n) = -\frac{p}{p+n} \log_2 \left(\frac{p}{p+n}\right) - \frac{n}{p+n} \log_2 \left(\frac{n}{p+n}\right) \tag{2.48}$$

denotes the information entropy and $E_{p(v)} \subset E_p$ and $E_{n(v)} \subset E_n$ are the subset of positive and negative examples, respectively, at this node for which attribute A assumes the value v. If the examples of a node belong to one class only, the node becomes a leaf node with the respective class. If no other attributes are left, the class of a leaf node is defined by the majority vote of the associated examples.

2.4 Maximum Entropy Models

The maximum entropy principle [31, 6] basically states that from a family of distributions which are consistent with our prior knowledge we should choose the one with the highest information entropy. The maximum entropy principle is quite general and we will restrict ourselves in this section to the case of estimating a finite, discrete distribution where the prior knowledge is given as a set of expectations.

Suppose we have a random variable X defined over the sample space $\mathcal{X} = \{x_1, \ldots, x_n\}$. Then the corresponding discrete distribution is parametrized by a n-dimensional probability vector $\mathbf{q} = (q_1, \ldots, q_n)$, which means $q_i \in [0, 1]$, and $\sum_i q_i = 1$, and $p(X = x_i) = q_i$. Among all possible distributions we seek the one with the highest entropy. If we have no further constraints, except for the normalization constraint, then we maximize

$$\hat{\mathbf{q}} = \arg\max_{\mathbf{q}} \mathbb{H}(\mathbf{q}) \qquad \text{such that} \sum_{i=1}^{n} q_i = 1 \tag{2.49}$$

$$= \arg\max_{\mathbf{q}} \sum_{i=1}^{n} -q_i \log q_i \tag{2.50}$$

where $\mathbb{H}(\mathbf{q})$ denotes the information entropy. We find that the uniform distribution with $\hat{q}_i = \frac{1}{n}$ for all i is the distribution with the highest entropy. Suppose, we are additionally given m arbitrary "feature" functions indexed by j

$$f_j : \mathcal{X} \to \mathbb{R} \qquad \text{for each } j = 1, \ldots, m. \tag{2.51}$$

For each of these functions we are given a value E_j that must be matched by the expected value of the function with respect to the discrete distribution, that is, it should hold that

$$E_j = \sum_i q_i f_j(x_i) \qquad \text{for each } j = 1, \ldots, m. \tag{2.52}$$

This way, each feature function along with its corresponding value E_j poses an additional constraint on the admissible values for the vector $\mathbf{q}$, because the functions f_j and the values E_j are fixed and our only hope to fulfill Eq. (2.52) is by finding the right values for the q_i. We thus have a constrained optimization problem: we seek to maximize the entropy $\mathbb{H}(\mathbf{q})$ subject to the m constraints of Eq. (2.52) plus one additional constraint which is the normalization constraint that ensures that the q_i sum to one. For each of these constraint we introduce a Lagrange multiplier $\lambda \in \mathbb{R}$ and transform the constrained optimization problem into an unconstrained optimization problem of an extended function $h(\mathbf{q}, \boldsymbol{\lambda})$ which now includes $\boldsymbol{\lambda}$ as a parameter and which has the following form

$$h(\mathbf{q}, \boldsymbol{\lambda}) = h(q_1, \ldots, q_n, \lambda_0, \lambda_1, \ldots, \lambda_m) \tag{2.53}$$

$$= \underbrace{-\sum_i q_i \log q_i}_{\text{entropy}} \underbrace{- \lambda_0 \left(\sum_i q_i - 1 \right)}_{\text{normalization constraint}} \underbrace{- \sum_j \lambda_j \left(E_j - \sum_i q_i f_j(x_i) \right)}_{\text{expectation constraints}}. \tag{2.54}$$

We will sometimes refer to the Lagrange multipliers $\lambda_1, \ldots, \lambda_m$ for the expectation constraints as "feature weights", the reason for this will soon become more apparent.

Now we need to find the parameters of $h(\mathbf{q}, \boldsymbol{\lambda})$ by setting its gradient to zero. The derivation for this is shown in Appendix A.2. This results in the following form for each q_i of the discrete distribution

$$q_i = \frac{\exp\left(\sum_j \lambda_j f_j(x_i)\right)}{\sum_{i'} \exp\left(\sum_j \lambda_j f_j(x_{i'})\right)}. \tag{2.55}$$

Thus, the probability $p(x_i) = q_i$ for each state x_i is proportional to an exponentiated sum of weighted feature functions where the $\boldsymbol{\lambda}_{1:m} = \lambda_1, \dots, \lambda_m$ play the role of feature weights. However, we still need to determine these weights. To do so, we consider the following scenario. Suppose, we have a set of n_s samples $\{x^{(s)}\}_{s=1}^{n_s}$ from the sample space $\mathcal{X}$ such that each $x^{(s)} \in \mathcal{X}$. This data set defines an empirical distribution

$$\tilde{p}(x) = \frac{1}{n_s} \sum_{s=1}^{n_s} \delta_{x^{(s)}}(x). \tag{2.56}$$

Further, suppose that the values E_j, that up to now were supposed to be arbitrary given values, are actually the feature expectations with respect to the empirical distribution $\tilde{p}(x)$ as represented by the data set, which means

$$E_j = \mathbb{E}_{\tilde{p}(x)}\left[f_j(x)\right] = \frac{1}{n_s} \sum_{s=1}^{n_s} f_j(x^{(s)}) \qquad \text{for each } j = 1, \dots, m. \tag{2.57}$$

Further, we suppose that these samples have been drawn from a distribution of the form as in Eq. (2.55). We can then consider a parameter estimation problem in a maximum likelihood setting. This means, we derive the feature weights by maximizing the log-likelihood of the data set with respect to the parameters, or, equivalently, the average log-likelihood $\ell(\boldsymbol{\lambda}_{1:m})$. Before we proceed, we introduce some notation to enhance readability. We denote by

$$\mathbf{f}(x) = (f_1(x), \dots, f_m(x)) \qquad \text{for any } x \in \mathcal{X} \tag{2.58}$$

the feature vector formed by the individual feature functions and we introduce a potential function $\psi(x)$ with

$$\psi(x) = \exp\left(\boldsymbol{\lambda}_{1:m} \mathbf{f}^{\mathsf{T}}(x)\right) \tag{2.59}$$

$$= \exp\left(\sum_j \lambda_j f_j(x)\right) . \tag{2.60}$$

Then we can rewrite Eq. (2.55) more compactly as

$$p(x) = \frac{\psi(x)}{\sum_{x'} \psi(x')} \tag{2.61}$$

where we now use the more common notion $\sum_x \psi(x)$ instead of $\sum_i \psi(x_i)$ – in both cases the sum is supposed to run over all $x \in \mathcal{X}$. Using this notation, the average log-likelihood $\ell(\boldsymbol{\lambda}_{1:m})$ of the data set with respect to the parameters is

$$\ell(\boldsymbol{\lambda}_{1:m}) = \frac{1}{n_s} \sum_s \log p(x^{(s)}) \tag{2.62}$$

$$= \frac{1}{n_s} \sum_s \log \frac{\psi(x^{(s)})}{\sum_x \psi(x)} \tag{2.63}$$

$$= \frac{1}{n_s} \sum_s \left(\log \psi(x^{(s)}) - \log \sum_x \psi(x)\right) . \tag{2.64}$$

The gradient of $\ell(\boldsymbol{\lambda}_{1:m})$ is

$$\nabla \ell(\boldsymbol{\lambda}_{1:m}) = \mathbb{E}_{\tilde{p}(x)}[\mathbf{f}(x)] - \mathbb{E}_{p(x)}\left[\mathbf{f}(x)\right] . \tag{2.65}$$

The derivation for the gradient is shown in Appendix A.3. Here, $\mathbb{E}_{\tilde{p}(x)}[\mathbf{f}(x)]$ is the expected feature vector with respect to the empirical distribution $\tilde{p}(x)$ as represented by the data set and $\mathbb{E}_{p(x)}\left[\mathbf{f}(x)\right]$ is the expected feature vector of the model using the current parameters. In short, the gradient is computed by the discrepancy between the empirical feature expectation and feature expectation of the model using the current parameters $\boldsymbol{\lambda}_{1:m}$.

The optimization problem is convex and the parameters will therefore approach a global optimum. It can be solved with any standard optimization method. For our approach presented in Chap. 4, we employ the RPROP algorithm [56] as an efficient gradient-based optimization technique.

2.5 Bayesian Nonparametrics

As noted in [34], the “basic idea of Bayesian nonparametrics is to replace classical finite-dimensional prior distributions with general stochastic processes”. In this section, we give a brief introduction to three stochastic processes that are commonly used in Bayesian nonparametrics. The first two are the Dirichlet process and the beta-Bernoulli process. In this context, we will also discuss hierarchical and nested extensions. The third and last process that we discuss is the Gaussian process as used in Gaussian process regression. For a more in-depth introduction to the first two processes, we also like to refer the interested reader to the tutorials of Teh and Jordan [70] and Gershman and Blei [23]. Gaussian process regression is presented in all detail in [54].

2.5.1 Dirichlet Process and Chinese Restaurant Process

The Dirichlet process can be considered as an infinite dimensional variant of the Dirichlet distribution and a draw from it can be considered as an infinite multinomial distribution. The Dirichlet process is frequently used as a prior in mixture models in which it replaces a finite dimensional Dirichlet *distribution*. To help grounding the theory, and for presenting a motivating example application, we will therefore first start with a discussion a finite Gaussian mixture models and the corresponding extension to a Dirichlet process Gaussian mixture model (DPGMM). We then give a formal definition of the Dirichlet process and, next, we illustrate an explicit representation of draws from this process by means of the stick-breaking construction. In contrast to parametric models, which have a finite number of parameters, the Dirichlet process is a nonparametric model with infinitely many parameters. This is problematic when we want to implement inference algorithms for models based on the Dirichlet process, as we obviously cannot explicitly represent infinitely many parameters. However, we will see that the posterior predictive distribution of the DP has a finite dimensional representation which permits feasible algorithms for posterior inference, for example in infinite mixture models. This posterior predictive distribution of a DP is closely related to the Chinese restaurant process (CRP). The connection to the DP is, that the sequential sampling of observations from the CRP can be described in terms of sampling from a sequence of posterior predictive distributions of a DP.

The description of the Dirichlet process and related concepts in this section is mainly based on [18, 47, 70, 71, 78] and we would like to redirect the interested reader to these articles for a more thorough description.

2.5.1.1 Finite Gaussian Mixture Models

The Dirichlet process is often used as a prior in mixture models and it will prove useful to keep this application in mind when we later discuss the Dirichlet process itself. In this section, we will therefore first start with a discussion of finite Gaussian mixture models (GMM) that make use of a Dirichlet *distribution* as a prior over the mixture weights and we will see that the use of a Dirichlet *process* prior will cause only a slight modification in the inference procedure. This change, however, will result in a more flexible model, which is able to adjust the number of mixture components during inference.

As a motivation, consider a scenario in which we are given a set of N points $\mathbf{x}_{1:N}$ with $\mathbf{x}_i \in \mathbb{R}^d$ for some $d \geq 1$ and we are interested in the predictive distribution $p(\mathbf{x}_* \mid \mathbf{x}_{1:N})$ that models the likelihood of observing a point at a certain location $\mathbf{x}_*$ given that we already have seen the points $\mathbf{x}_{1:N}$. We assume that these points have been drawn from a Gaussian mixture density $p(\mathbf{x} \mid \boldsymbol{\beta})$ parametrized by some $\boldsymbol{\beta}$. The parameters $\boldsymbol{\beta}$ include the mixture weights and the parameters of the Gaussians, but we will postpone these details for the moment. The actual form of the mixture density is unknown to us but we have a prior distribution $p(\boldsymbol{\beta})$ over possible mixture densities. By observing the data points our prior over mixture densities is updated and we have a more informed posterior distribution over mixture densities

$$p(\boldsymbol{\beta} \mid \mathbf{x}_{1:N}) \propto p(\boldsymbol{\beta}) \prod_{i=1}^{N} p(\mathbf{x}_i \mid \boldsymbol{\beta}). \tag{2.66}$$

By marginalizing the mixture density we arrive at the posterior predictive distribution

$$p(\mathbf{x}_* \mid \mathbf{x}_{1:N}) = \int_{\boldsymbol{\beta}} p(\mathbf{x}_* \mid \boldsymbol{\beta}) p(\boldsymbol{\beta} \mid \mathbf{x}_{1:N}) \, d\boldsymbol{\beta} \tag{2.67}$$

which models the likelihood of a new point at location $\mathbf{x}_*$ given the previously seen points. The trouble is, we might not be able solve the integral analytically. However, we can use Markov chain Monte Carlo techniques to obtain samples $\{\boldsymbol{\beta}^{(j)}\}_{j=1}^{M}$ from the posterior distribution $p(\boldsymbol{\beta} \mid \mathbf{x}_{1:N})$ and compute a Monte Carlo approximation of the posterior predictive distribution as

$$p(\mathbf{x}_* \mid \mathbf{x}_{1:N}) \approx \frac{1}{M} \sum_{j=1}^{M} p(\mathbf{x}_* \mid \boldsymbol{\beta}^{(j)}). \tag{2.68}$$

In the following, we describe the details of how to utilize Gibbs sampling to draw samples from the posterior distribution. We are given N data points $\mathbf{x}_{1:N}$ with $\mathbf{x}_i \in \mathbb{R}^d$ that have been sampled from a mixture density consisting of K Gaussians

$$\mathbf{x}_i \sim p(\cdot \mid \mathbf{w}, \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1, \ldots, \boldsymbol{\mu}_K, \boldsymbol{\Sigma}_K) = \sum_{k=1}^{K} w_k \mathcal{N}(\cdot \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k). \tag{2.69}$$

This mixture density is parametrized by the mixture weights $\mathbf{w} = (w_1, \ldots, w_K)$, which sum to one, and the parameters of the Gaussians, i.e., their respective mean $\boldsymbol{\mu}_k$ and covariance matrix $\boldsymbol{\Sigma}_k$. We will sometimes refer to the Gaussians as mixture components or clusters and refer to their parameters as cluster parameters $\boldsymbol{\theta}_k = (\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$. We are free to choose any suitable prior for the latent parameters of the mixture density. In our example, we will use their respective conjugate priors and thus place a Dirichlet prior $p(\mathbf{w})$ on the weights and a normal-Wishart prior $p(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ on each of the cluster parameters.

Sampling data points from the implied generative process works as follows. First, we draw a mixture density by sampling K Gaussians from the normal-Wishart distribution and the mixture weights from a Dirichlet distribution. Given the resulting mixture density, we draw each data point $\mathbf{x}_i$ independently by first sampling a cluster component $c_i \sim p(c_i \mid \mathbf{w})$ according to the weights. Then, we sample the actual observable data point $\mathbf{x}_i \sim \mathcal{N}(\mathbf{x}_i \mid \boldsymbol{\mu}_{c_i}, \boldsymbol{\Sigma}_{c_i})$ from the corresponding Gaussian c_i. Please note, that we thereby introduced correspondence variables $c_i \in \{1, \ldots, K\}$ that denote the cluster from which a point has been sampled from. For example, if $c_4 = 3$ then point $\mathbf{x}_4$ has been sampled from cluster 3. The correspondence variables are also treated as latent variables. In accordance with the introductory remarks, we denote the latent variables by $\boldsymbol{\beta}$ but we now include the correspondence variables, so we have $\boldsymbol{\beta} = (\mathbf{w}, \boldsymbol{\theta}_{1:K}, \mathbf{c}_{1:N})$.

To wrap up, the generative process can be described more concisely as

$$\begin{aligned}
\mathbf{w} = (w_1, \ldots, w_K) &\sim \mathrm{Dir}(\cdot \mid \boldsymbol{\alpha}) && \text{once} && (2.70)\\
\boldsymbol{\theta}_k = (\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) &\sim \mathcal{NW}(\cdot \mid \nu_0, \kappa_0, \boldsymbol{\mu}_0, \mathbf{T}_0) && \text{for } k = 1, \ldots, K && (2.71)\\
c_i &\sim \mathrm{Mult}(\cdot \mid \mathbf{w}) && \text{for } i = 1, \ldots, N && (2.72)\\
\mathbf{x}_i &\sim \mathcal{N}(\cdot \mid \boldsymbol{\mu}_{c_i}, \boldsymbol{\Sigma}_{c_i}) && \text{for } i = 1, \ldots, N && (2.73)
\end{aligned}$$

and the corresponding Bayesian network is depicted in Fig. 2.3.

A complete Gibbs sweep samples new values for each of the latent variables from their respective conditional distributions. We start with an arbitrary initial state $\boldsymbol{\beta}^{(0)}$. It will prove useful to have an intuitive visualization of this state. If, as a toy example, we would

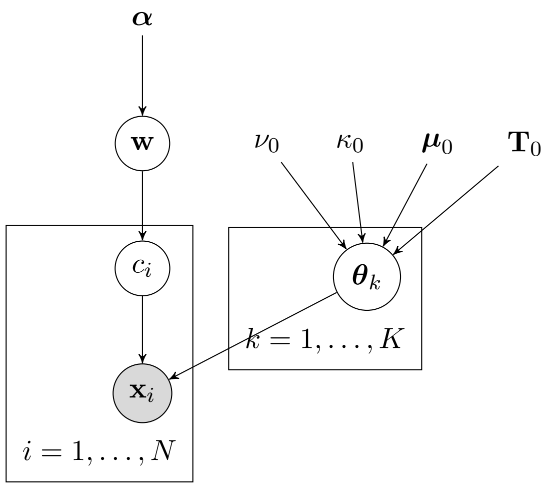


Figure 2.3: Bayesian network representing a finite Gaussian mixture model.

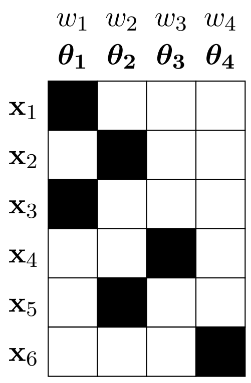


Figure 2.4: A visual representation of a sample $\boldsymbol{\beta}^{(i)}$.

have four Gaussians and six data points then a state $\boldsymbol{\beta}^{(i)}$ of the Markov chain could be visualized by the matrix depicted in Fig. 2.4. Here, each row corresponds to a data point and each column corresponds to a cluster. The binary matrix visualizes the correspondence variables, e.g. here the data point $\mathbf{x}_4$ is currently associated to the third Gaussian which means we have $c_4 = 3$. Further, each column is associated with the cluster variables: the parameters $\boldsymbol{\theta}_k$ of a Gaussian and its weight w_k. We now need to resample the entries of the binary matrix, i.e., the correspondence variables, as well as the cluster parameters. For the correspondence variable we sample a new value from the following distribution

$$\begin{aligned} p(c_i \mid \mathbf{x}_{1:N}, \mathbf{c}_{[-i]}, \mathbf{w}, \boldsymbol{\theta}_{1:K}) &= p(c_i \mid \mathbf{x}_i, \mathbf{w}, \boldsymbol{\theta}_{1:K}) && (2.74) \\ &\propto p(\mathbf{x}_i \mid c_i, \boldsymbol{\theta}_{1:K}) p(c_i \mid \mathbf{w}) && (2.75) \\ &= \underbrace{\mathcal{N}(\mathbf{x}_i \mid \boldsymbol{\mu}_{c_i}, \boldsymbol{\Sigma}_{c_i})}_{\text{likelihood}} \underbrace{w_{c_i}}_{\text{prior}} . && (2.76) \end{aligned}$$

As can be seen in Eq. (2.76) the probability of assigning data point $\mathbf{x}_i$ to cluster c_i depends on the cluster's prior probability (which is determined by the current weight w_{c_i} of the cluster) and the likelihood under the cluster's data distribution (which is the likelihood of the point $\mathbf{x}_i$ under the current parametrization of the cluster's Gaussian distribution).

To complete one round of Gibbs sampling, we would resample the weights and parameters of the Gaussians. But there is an alternative to this. Remember that our initial motivation was to integrate out the model parameters. We then resorted to a Monte Carlo approximation of the integral, because we realized we could not solve the integral analytically in general. However, because we used conjugate priors for the mixture weights and the cluster parameters, we actually can integrate out at least these variables, though we still need to explicitly represent and sample the correspondence variables. Thus, the posterior over the latent variables can be expressed solely in terms of the correspondence variables $p(\mathbf{c}_{1:N} \mid \mathbf{x}_{1:N})$. This modified Gibbs sampling procedure is known as a collapsed Gibbs sampler. It will resample each correspondence variable c_i from the following distribution

$$p(c_i \mid \mathbf{c}_{[-i]}, \mathbf{x}_{1:N}) \propto \underbrace{p(\mathbf{x}_i \mid \mathbf{c}, \mathbf{x}_{[c_i]})}_{\text{likelihood}} \underbrace{p(c_i \mid \mathbf{c}_{[-i]})}_{\text{prior}} . \qquad (2.77)$$

Here, $\mathbf{x}_{[c_i]}$ denotes all data points currently associated to the the cluster c_i. As can be seen, the marginalization of the mixture weights and cluster parameters introduced new dependencies between some of the variables which previously have been conditionally independent.

For example, the conditional probability $p(c_i \mid \mathbf{w})$ of a correspondence variable was independent of other correspondence variables given the weights. By marginalizing the weights, the correspondence variables are now dependent on each other, which results in the following new form of the cluster prior

$$p(c_i = k \mid \mathbf{c}_{[-i]}) = \frac{n_k + \alpha_k}{\sum_j n_j + \alpha_j} . \qquad (2.78)$$

Here, n_k is the number of points currently assigned to cluster k, where the count excludes the association of point $\mathbf{x}_i$ itself. In the matrix representation of Fig. 2.4, this corresponds to the number of black squares in column k, where we ignore the row of point $\mathbf{x}_i$. The α_k are parameters of the Dirichlet prior which in this case act as pseudo-counts for the K clusters.

Similar arguments hold for the likelihood $p(\mathbf{x}_i \mid c_i, \boldsymbol{\theta}_{1:K})$ of a single data point which was independent of other data points given its correspondence variable c_i and the cluster parameters $\boldsymbol{\theta}_{1:K}$. By marginalizing out the cluster parameters the likelihood now becomes dependent on all other data points $\mathbf{x}_{[c_i]}$ currently assigned to cluster c_i

$$p(\mathbf{x}_i \mid \mathbf{c}, \mathbf{x}_{[c_i]}) = t_v(\mathbf{x}_i \mid \boldsymbol{\mu}_{c_i}, \boldsymbol{\Sigma}_{c_i}). \tag{2.79}$$

This is the posterior predictive distribution with respect to a normal-Wishart prior. The resulting distribution is a multivariate t-Distribution $t_v(\mathbf{x}_i \mid \boldsymbol{\mu}_{c_i}, \boldsymbol{\Sigma}_{c_i})$, where the parameters $\boldsymbol{\mu}_{c_i}$ and $\boldsymbol{\Sigma}_{c_i}$ are determined both by the parameters $\nu_0, \kappa_0, \boldsymbol{\mu}_0, \mathbf{T}_0$ of the normal-Wishart prior as well as the location of the data points $\mathbf{x}_{[c_i]}$. The details for calculating this likelihood can be found in the appendix A.1. To summarize, the collapsed Gibbs sampler resamples the cluster assignments from the following distribution

$$p(c_i = k \mid \mathbf{c}_{[-i]}, \mathbf{x}_{1:N}) \propto t_v(\mathbf{x}_i \mid \boldsymbol{\mu}_{c_i}, \boldsymbol{\Sigma}_{c_i})(n_k + \alpha_k). \tag{2.80}$$

2.5.1.2 Dirichlet Process Gaussian Mixture Models

A drawback of finite Gaussian mixture models is that we have to specify the number of clusters K in advance. It is unlikely that in any practical setting we could assume to know this number and we should rather treat it as another latent variable. Hence, it would be desirable to have a more flexible model that could increase or decrease the number of instantiated mixture components during inference as needed. The Gibbs sampling procedure would then provide us with samples that additionally imply a posterior distribution over the number of mixture components.

Such a model can be obtained by replacing the Dirichlet *distribution* over the mixture weights with a Dirichlet *process* and the resulting model is called a Dirichlet process Gaussian mixture model (DPGMM) [47]. The usage of a Dirichlet process prior reflects the assumption that there are actually infinitely many mixture components. Accordingly, if we draw the mixture weights from a Dirichlet process we get a vector with infinitely many entries that still sum to one. As for the finite variant, we can also analytically integrate out the weight vector and the resulting posterior predictive distribution for a correspondence variable c_i has a similarly simple form. Denoting the number of currently instantiated clusters in the model by K, the new cluster prior is

$$p(c_i = k \mid \mathbf{c}_{[-i]}) = \begin{cases} \frac{n_k}{\alpha + \sum_j n_j} & k \text{ is an existing cluster } (k \leq K) \\ \frac{\alpha}{\alpha + \sum_j n_j} & k \text{ is a new cluster } (k = K + 1) \end{cases} . \tag{2.81}$$

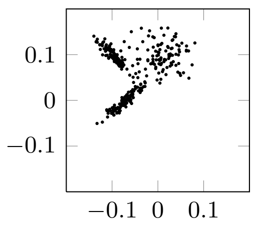
Here, α is a parameter stemming from the Dirichlet process. The prior to assign a data point to an already existing cluster is proportional to the number n_k of points already associated with this cluster (not counting the point $\mathbf{x}_i$ itself). With a probability proportional to α we would add a new cluster to our model and the point will be associated to this newly instantiated cluster. This would correspond to the expansion of the matrix in Fig. 2.4 by one column. If $\mathbf{x}_i$ is the only point of its cluster, then this cluster is removed from the model prior to resampling the correspondence variable c_i. Please note, that K is not a predefined constant anymore, rather it will be implicitly determined by the correspondence variables.

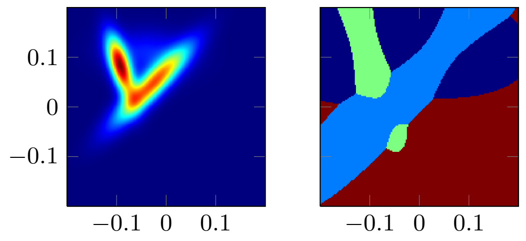
For the complete conditional distribution that is used by the Gibbs sampler, we only need to combine this prior with the likelihood of the data distribution of the respective clusters

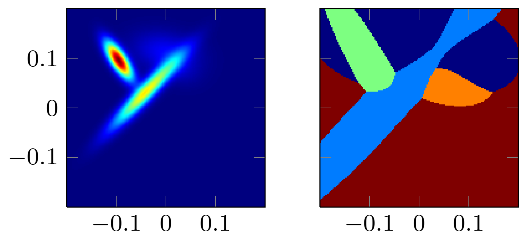
$$p(c_i = k \mid \mathbf{c}_{[-i]}, \mathbf{x}_{1:N}) \propto \begin{cases} p(\mathbf{x}_i \mid \mathbf{c}, \mathbf{x}_{[k]})\, n_k & k \text{ is an existing cluster } (k \leq K) \\ p(\mathbf{x}_i \mid \mathbf{c})\, \alpha & k \text{ is a new cluster } (k = K + 1) \end{cases} . \tag{2.82}$$

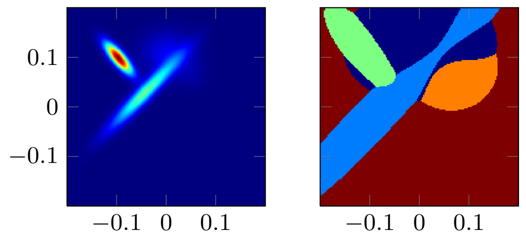
Here, we assume that the cluster parameters are integrated out and we work with the posterior predictive distributions $p(\mathbf{x}_i \mid \mathbf{c}, \mathbf{x}_{[k]})$ with respect to the normal-Wishart prior. When considering a new cluster, the likelihood is the prior predictive distribution $p(\mathbf{x}_i \mid \mathbf{c})$ of the normal-Wishart prior as there are no points associated to this newly instantiated cluster yet.

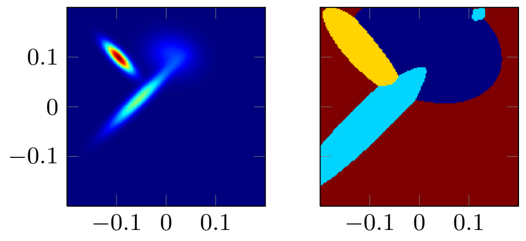
In Fig. 2.5 we illustrate an example run of a collapsed Gibbs sampler for a DPGMM. Each iteration corresponds to one Gibbs sweep during which each correspondence variable is sampled once. For generating the data set that is shown on top, we sampled 300 data points from three Gaussians. The Dirichlet process prior uses a concentration parameter of $\alpha = 1.0$ and the normal-Wishart prior has parameters $\boldsymbol{\mu}_0 = (0,0)^\mathsf{T}, \kappa_0 = 0.2, \nu_0 = 5.0$, and $\mathbf{T}_0 = 0.0025\mathbf{I}$. In the example run shown in Fig. 2.5, we had either three or four active clusters in each iteration, though we also saw more clusters in different runs. If more than three cluster existed, then these additional clusters usually had no more than a few data points assigned to it. For each iteration shown in Fig. 2.5, we depict on the left side the posterior predictive distribution of the mixture density when conditioned on the data points

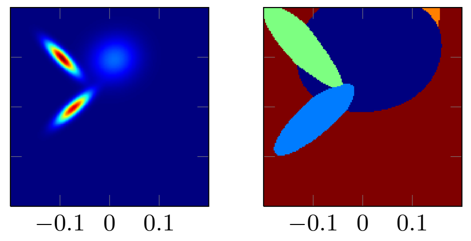


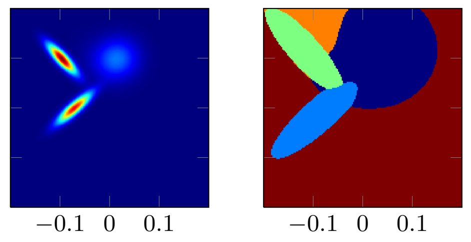

(a) Iteration 1.

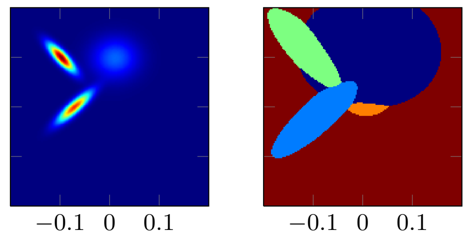

(b) Iteration 2.


(c) Iteration 3.


(d) Iteration 5.


(e) Iteration 10.


(f) Iteration 15.


(g) Iteration 20.

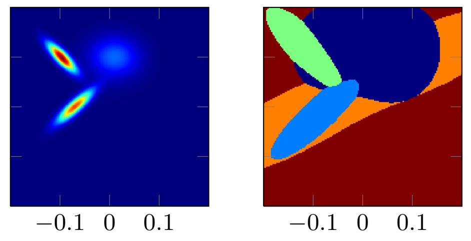

(h) Iteration 25.

Figure 2.5: Example run of a collapsed Gibbs sampler for a DPGMM.

with their current cluster assignments. On the right side, we depict the most likely cluster assignment for a new point. Any new point that falls into the red area would most likely be assigned to a newly instantiated cluster.

We have seen, that this sampling technique can increase or decrease the number of components in the mixture model, which allows the effective model complexity to adapt to the data complexity as needed. Hence, this gives us the desired flexibility of not having to specify the number of components in advance. We now rather have a prior over the number of components which is influenced by the parameter α. The higher α the more components we expect a priori. Posterior inference in this model remains tractable, even though we assume that there exist infinitely many mixture components, because we only have to consider the currently instantiated clusters plus the possibility of expanding the model with one additional cluster. This makes the model also appealing from a computational perspective. In the following, we will discuss the Dirichlet process in more detail.

2.5.1.3 Definition of the Dirichlet Process

The Dirichlet process was introduced by Ferguson [18]. It can be defined as follows.[1] Let G_0 be a distribution over the sample space Ω, and let $\alpha > 0$ be a positive real number. Then a random distribution G over the sample space Ω is distributed according to a Dirichlet process with concentration parameter α and base distribution G_0

$$G \sim \mathrm{DP}(\alpha, G_0), \tag{2.83}$$

if for all $k \geq 1$ and all partitions $(A_1, \dots, A_k)$ of Ω the vector $(G(A_1), \dots, G(A_k))$ is Dirichlet distributed according to

$$(G(A_1), \dots, G(A_k)) \sim \mathrm{Dir}(\alpha G_0(A_1), \dots, \alpha G_0(A_k)). \tag{2.84}$$

The intuition behind Eq. (2.84) will become more apparent in the next section when we look at an explicit representation for a draw G.

2.5.1.4 Stick-breaking Construction

The definition of the Dirichlet process as given above was not a constructive one. It did not provide us with any means to actually draw a distribution G that satisfies these require-

[1]This is a slightly simplified definition that avoids measure theoretic jargon.

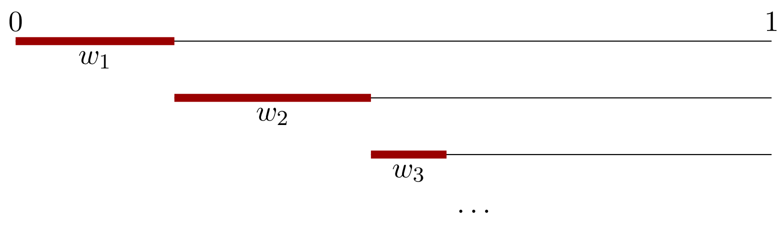


Figure 2.6: Stick-breaking construction for the Dirichlet process.

ments. In the following this will be remedied by the so-called stick-breaking construction.

A draw $G \sim \mathrm{DP}(\alpha, G_0)$ from a Dirichlet process can be explicitly represented as an infinite sum of weighted Dirac functions centered at values θ_i

$$G(\cdot) = \sum_{i=1}^{\infty} w_i \delta_{\theta_i}(\cdot). \tag{2.85}$$

Hence, a sample G from the Dirichlet process is parametrized by an infinite set of tuples $\{(w_i, \theta_i)\}_{i=1}^{\infty}$. To draw a sample form a DP, we need to sample the individual values w_i and θ_i such that the corresponding G is distributed according to a Dirichlet process. The w_i and θ_i are sampled independently of each other. Let us first consider the atoms θ_i, which are simply sampled independently from the base distribution G_0

$$\theta_i \sim G_0 \qquad \text{for } i = 1, \ldots, \infty \tag{2.86}$$

Next, the weights w_i are sampled according to

$$v_i \sim \mathrm{Beta}(1, \alpha) \qquad \text{for } i = 1, \ldots, \infty \tag{2.87}$$

$$w_i = v_i \prod_{k=1}^{i-1} (1 - v_k) \qquad \text{for } i = 1, \ldots, \infty \tag{2.88}$$

where α in Eq. (2.87) is the concentration parameter of the Dirichlet process. This is the so-called stick-breaking process and there is an intuitive metaphor for describing this process. One imagines a stick of length one and then breaks off a segment at a proportion sampled from $\mathrm{Beta}(1, \alpha)$. This will be the weight w_1 for the first component of the infinite sum. The remaining stick has length $1 - w_1$ and is then treated similarly. This process recurses infinitely. Because the stick had a length of one, the infinitely many weights, which correspond to the segments of the stick, must sum to one (see also Fig. 2.6).

The intuition behind Eq. (2.84) should now be more apparent. Remember that the atoms of G have been sampled from the base distribution G_0 which is defined over the space Ω and $(A_1, \ldots, A_k)$ forms a partition of this space. Thus, each atom of G belongs to exactly one subset A_i of Ω. The probability mass $G(A_i)$ assigned to subset A_i by the distribution G is simply the sum of the weights of the atoms within this region. Hence, the vector $(G(A_1), \ldots, G(A_k))$ is a probability vector, i.e., its entries are in the interval $[0, 1]$ and they sum to one. If we keep the partition fixed and repeatedly sample G, then we would get different probability vectors, because the weights and locations of the atoms changed. Thus, for a given partition of Ω the Dirichlet process implicitly defines a distribution over the corresponding probability vectors – and these vectors are distributed according to a Dirichlet distribution with parameters $\alpha G_0(A_i), \ldots, \alpha G_0(A_k)$ and hence

$$(G(A_1), \ldots, G(A_k)) \sim \mathrm{Dir}(\alpha G_0(A_1), \ldots, \alpha G_0(A_k)). \tag{2.89}$$

2.5.1.5 Chinese Restaurant Process

If we draw samples from G as given in Eq. (2.85), we obtain the atom θ_i with probability w_i. Now suppose we already have obtained a set of samples $\theta^{(1)}, \ldots, \theta^{(n)}$ from G with $\mathrm{DP}(\alpha, G_0)$ being the Dirichlet process prior over G. Then, the predictive distribution over a new atom θ given the previous samples and the DP prior arises when we integrate out G. This leads to:

$$\begin{aligned} p(\theta \mid \theta^{(1)}, \ldots, \theta^{(n)}, G_0, \alpha) &= \frac{\alpha}{\alpha + n} G_0(\theta) + \frac{1}{\alpha + n} \sum_{i=1}^{n} \delta_{\theta^{(i)}}(\theta) && (2.90) \\ &= \frac{\alpha}{\alpha + n} G_0(\theta) + \frac{n_\theta}{\alpha + n}. && (2.91) \end{aligned}$$

Here, n_θ is the number of times the atom θ has been previously sampled.

This distribution is closely connected to the Chinese restaurant process (CRP). In this metaphor, one imagines customers arriving sequentially at a Chinese restaurant with an infinite number of tables. The first customer is seated at one of the tables and samples an atom $\theta \sim G_0$ from the base distribution. This will be the parameter of the table and any subsequent customer seated at this table will inherit this parameter. When the n-th customer arrives she will sit at an already occupied table with a probability proportional to the number of customers sitting at this table and with a probability proportional to α she will sit at a new table. In the second case, she also draws the parameter for this newly occupied table. When the $(n+1)$-th customer is seated, the parameter θ at her table appears to be sampled from the posterior predictive distribution in Eq. (2.90). Note, that the table assignments also imply a clustering of the customers.

2.5.1.6 Dirichlet Process Gaussian Mixture Models Revisited

We will now shortly revisit the Gaussian mixture model, to illustrate how the Dirichlet process and the Chinese restaurant process are linked to the Dirichlet process Gaussian mixture model (DPGMM) and the Gibbs sampling procedure that we described above.

Consider the following model which specifies a a particular multivariate DPGMM from which we draw n points $\mathbf{x}_i$

$$G \sim \mathrm{DP}(\alpha, \mathcal{NW}(\nu_0, \kappa_0, \boldsymbol{\mu}_0, \mathbf{T}_0)) \qquad \text{once} \qquad (2.92)$$

$$\{\boldsymbol{\mu}_{c_i}, \boldsymbol{\Lambda}_{c_i}\} \sim G \qquad \text{for } i = 1, \ldots, n \qquad (2.93)$$

$$\mathbf{x}_i \sim \mathcal{N}(\boldsymbol{\mu}_{c_i}, \boldsymbol{\Lambda}_{c_i}^{-1}) \qquad \text{for } i = 1, \ldots, n \qquad (2.94)$$

In Eq. (2.92) we draw G from a Dirichlet process with a concentration parameter α and a normal-Wishart prior $\mathcal{NW}$ that plays the role of the base distribution. Remember that G is a set of infinitely many weighted atoms $\{w_j, \theta_j\}_{j=1}^{\infty}$, where the weights are drawn according to the stick-breaking process that is influenced by the concentration parameter α, while the atoms θ_j are sampled independently from the base distribution. In our case, the base distribution is a normal-Wishart distribution, which is a distribution over Gaussians that are parametrized by their respective means $\boldsymbol{\mu}_j$ and precision matrices $\boldsymbol{\Lambda}_j$. Thus, each atom $\theta_j = \{\boldsymbol{\mu}_j, \boldsymbol{\Lambda}_j\}$ in G corresponds to a Gaussian distribution and G corresponds to an infinite Gaussian mixture model consisting of infinitely many weighted Gaussians with different means and covariance matrices. Next, in Eq. (2.93) we draw one Gaussian mixture component from G proportional to its weight. Here, c_i is the correspondence variable of the i-th point and denotes the index of the Gaussian we just sampled. The actual observable data point $\mathbf{x}_i$ is then sampled in Eq. (2.94) from the corresponding Gaussian distribution.

Instead of first drawing G from the Dirichlet process and then drawing the points independently from the Gaussians defined by G we could also follow another sampling procedure by switching to the Chinese restaurant representation. In this case, we would draw the n points from a sequence of posterior predictive distribution $p(\mathbf{x}_i \mid \mathbf{x}_1, \ldots, \mathbf{x}_{i-1})$ that arise when we integrate out the draw G from the Dirichlet process. Following the Chinese restaurant metaphor, the data points correspond to customers, the mixture components correspond to the tables and each table is associated with a table parameter which is a Gaussian drawn from the normal-Wishart prior. Drawing a data point in this representation works as follows. First, a customer is seated according to the usual CRP rules at one of the tables. Whenever a customer is seated at a new table we once draw a Gaussian from the normal-Wishart prior and this Gaussians then becomes associated with this table and is reused for any subsequent customers sitting at this table. The observed data point $\mathbf{x}_i$ is then drawn from the Gaussian at the table where it has been seated.

This Chinese restaurant representation of the above-defined DPGMM finally provides the link to the Gibbs sampling procedure described earlier. In this procedure, we were interested in inverting the generative process by inferring the cluster or table assignments c_i of a given set of data points. If we also integrate out the table parameters, i.e. the Gaussians, then this corresponds to the collapsed Gibbs sampler.

2.5.2 Beta-Bernoulli Process and Indian Buffet Process

In the previous sections we saw that mixture models based on the Dirichlet process or the Chinese restaurant process are useful for clustering data into distinct groups where each data point is associated with one out of several latent clusters. For other applications, it might be more appropriate to model data points or observations in terms of features, where an observation would be associated with several features instead of a single class or cluster. The task would then be to infer the latent features themselves and, for each observation, a binary feature vector $\mathbf{d}_i = (d_{i,1}, \dots, d_{i,k})$ with $d_{i,j} \in \{0, 1\}$ that indicates for each of the k features whether the observation is associated with it or not.

In a Bayesian setting, we need a prior for the feature vector of each individual observation. If we assume that the features are independent we can use a Bernoulli distribution as a prior for the presence or absence of a certain feature in a given observation. Further, we could use a single beta distribution as a prior over the parameters of the k Bernoulli distributions. This is a finite model in which we have to specify a fixed number of features in advance. For a more flexible model, we could assume that there exist infinitely many features. For this, we replace the beta distribution with a beta process, which acts as a prior for the parameters of an infinite collection of Bernoulli variables.

The beta process $\mathrm{BP}(c, G_0^\star)$ is parametrized by a concentration parameter c and a base measure $G_0^\star$ that puts total mass $\alpha = G_0^\star(\Omega)$ on the sample space Ω. To make the mass parameter α more obvious, we will use the notation $\mathrm{BP}(c, \alpha, G_0)$, where $G_0 = \frac{1}{\alpha} G_0^\star$ is a probability measure or base distribution. A draw

$$G \sim \mathrm{BP}(c, \alpha, G_0) \tag{2.95}$$

from this process can be represented as an infinite set $\{w_i, \theta_i\}_{i=1}^{\infty}$ of weighted atoms drawn from the base distribution G_0. In the Dirichlet process, the weights modeled a discrete distribution and the weights therefore sum to one. In contrast, a draw from the beta process can be thought of as an infinite collection of Bernoulli variables where their parameters are defined by the weights w_i of the atoms. Hence, G parametrizes a Bernoulli process and if we sample from this process

$$\boldsymbol{\theta} = \{\theta_1, \ldots, \theta_n\} \sim \mathrm{BeP}(G) \tag{2.96}$$

we get a *set* of atoms, where atom θ_i is part of the collection with probability w_i and with probability $1 - w_i$ it is absent.[2]

There also exist different stick-breaking constructions for the beta process. One variant [72], which is defined only for the special case of $c = 1$, has an interesting connection to the stick-breaking construction of the Dirichlet process. It is defined as follows

$$\theta_i \sim G_0 \qquad \text{for } i = 1, \ldots, \infty \tag{2.97}$$

$$v_i \sim \mathrm{Beta}(1, \alpha) \qquad \text{for } i = 1, \ldots, \infty \tag{2.98}$$

$$w_i = \prod_{k=1}^{i} (1 - v_k) \qquad \text{for } i = 1, \ldots, \infty \tag{2.99}$$

In Eq. (2.97), the location θ_i of the atoms are drawn independently from the base distribution G_0. In Eq. (2.98), the break points v_i for the stick are sampled just as for the Dirichlet process, but in Eq. (2.99) the weights are defined to be the other part of the stick – the part from which we resume to break off segments. This is also illustrated in Fig. 2.7 on the next page. Without going into details, we like to point out that a stick-breaking construction for

[2]For the sake of simplicity, we present the draws from the beta process and the Bernoulli process not in terms of random measures but in terms of the corresponding random parameters that parametrize these measures. We sometimes also have done this when discussing the Dirichlet process. For example, a draw $G \sim \mathrm{DP}$ from the Dirichlet process is a probability measure $G(\cdot) = \sum_{i=1}^{\infty} w_i \delta_{\theta_i}(\cdot)$ with parameters $\{w_i, \theta_i\}_{i=1}^{\infty}$. A draw $H \sim \mathrm{BP}$ from the beta process is a completely random measure $H(\cdot) = \sum_{i=1}^{\infty} w_i \delta_{\theta_i}(\cdot)$ with parameters $\{w_i, \theta_i\}_{i=1}^{\infty}$. A draw $Z \sim \mathrm{BeP}$ from the Bernoulli process is a completely random measure $Z(\cdot) = \sum_{i=1}^{\infty} d_i \delta_{\theta_i}(\cdot)$ with parameters $\{d_i, \theta_i\}_{i=1}^{\infty}$ with $d_i \in \{0, 1\}$ and $d_i \sim \mathrm{Bern}(w_i)$ with w_i stemming from the corresponding $H \sim \mathrm{BP}$. Hence, the measure Z could be parametrized solely in terms of the finite set of "active" atoms $\{\theta_k\}_{k=1}^{n}$ for which $d_k = 1$ as done in Eq. (2.96). For details see also [49, 73].

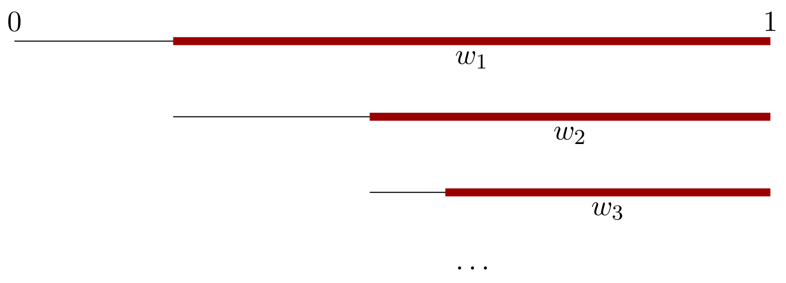


Figure 2.7: An illustration of the stick-breaking construction for the beta process for the special case of $c = 1$. Contrary to the construction of the DP, the weights are now defined to be the other part of the stick from which we continue to break off segments.

the general case of $c > 0$ is described in [49].

In practice, the beta-Bernoulli process is augmented with an application specific data distribution $p(x \mid \boldsymbol{\theta})$, which is now conditioned on a set of atoms $\boldsymbol{\theta} = \{\theta_1, \ldots, \theta_n\}$ instead of just a single atom as was the case for mixture models based on the Dirichlet process. To draw n data points from such a model, we would proceed as follows

$$G \sim \mathrm{BP}(c, \alpha, G_0) \qquad \text{once} \qquad (2.100)$$
$$\boldsymbol{\theta}^{(i)} = \{\theta^{(i,1)}, \ldots, \theta^{(i,m)}\} \sim \mathrm{BeP}(G) \qquad \text{for } i = 1, \ldots, n \qquad (2.101)$$
$$x_i \sim p(\cdot \mid \boldsymbol{\theta}^{(i)}) \qquad \text{for } i = 1, \ldots, n \qquad (2.102)$$

For a given set of data points $\{x_i\}_{i=1}^n$, we would be interested to invert this generative process to infer the latent features $\{\theta_1, \theta_2, \ldots\}$. Further, we would like to infer for each data point x_i the set of active features $\boldsymbol{\theta}^{(i)}$ that have been used to generate the corresponding point from the data distribution in Eq. (2.102). For this, we could introduce the above-mentioned feature vector $\mathbf{d}_i = (d_{i,1}, \ldots, d_{i,k})$ and use a Gibbs sampling procedure to draw samples from the posterior distribution over the latent variables given the observed data points. However, it is impractical to represent G in a MCMC-based inference procedure, because G has infinitely many parameters. For mixture models based on the Dirichlet process this has been resolved by analytically integrating out the draw from the DP which gave rise to the Chinese restaurant representation. The analog for the beta-Bernoulli process is the Indian buffet process (IBP) [24], which arises when we integrate out the draw G from the beta process. While in the CRP a customer has been associated to a single class (or table), in the IBP a customer would be associated to multiple features (or dishes).

More specifically, in the IBP metaphor one imagines customers arriving sequentially at an Indian buffet with an infinite number of dishes where each dish is associated with a dish parameter θ sampled from the base distribution G_0. The first customer at the buffet tastes a certain number of dishes and this number $n_{1,+}$ is drawn from a Poisson distribution with mean α

$$n_{1,+} \sim \text{Pois}\left(\alpha\right), \tag{2.103}$$

where α is the mass parameter of the beta process. The i-th customer arriving at the buffet tastes some of the dishes that previous customers have tasted already, as well as a certain number of new dishes. The previously chosen dishes are selected based on their popularity, that is, with a probability of

$$\frac{n_k}{i-1+c} \tag{2.104}$$

the i-th customer also tastes dish k, where $n_k \geq 1$ is the number of customers that previously selected dish k. Here, c is the concentration parameter of the beta process. This decision is made independently for each of the previously tasted dishes, which reflects the fact that the atoms of G model independent Bernoulli variables. The i-th customer then additionally tastes a Poisson distributed number $n_{i,+}$ of new dishes not chosen by any customer before her

$$n_{i,+} \sim \text{Pois}\left(\frac{c\alpha}{i-1+c}\right). \tag{2.105}$$

This also includes the possibility of not choosing any new dishes at all. Whenever a dish has been tasted for the first time, we once sample its associated dish parameter θ from the base distribution G_0. This is the two-parameter IBP introduced in [73]. The original IBP [24] is recovered with $c = 1$. Further, we like to point out that a three parameter generalization of the IBP has been proposed in [69].

We could then use the IBP representation for a Gibbs sampler that resamples the dish assignments for each customer. As with the CRP, during Gibbs sampling one can assume that any customer is the last one to arrive at the buffet and resample the dish assignments accordingly. The data likelihood of a customer (data point) then depends on the associated dish parameters of all dishes she has tasted. Similar as for the CRP, a state of the Gibbs sampler can be visualized by the corresponding association matrix shown in Fig. 2.8 on the next page. The rows correspond to the dishes of the buffet and the rows to the customers. Please note, that in contrast to the CRP, a row may now contain multiple black squares (associations). If the data distribution is conjugate to the base distribution of the beta process, then the dish parameters can be integrated out analytically and do not have to be explicitly represented during Gibbs sampling. However, depending on the application, for the IBP

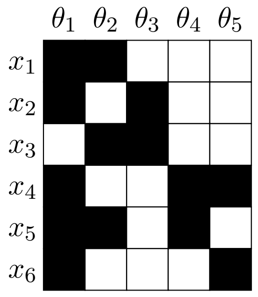


Figure 2.8: An illustration of an Indian buffet process. The columns correspond to the dishes and the rows to the customers. The matrix depicts the customer to dish associations. The $\theta_i \sim G_0$ are the dish parameters sampled from the base distribution G_0 and the x_i are the data points. The data likelihood $p(x_i \mid \boldsymbol{\theta})$ of a point x_i depends on the set of features $\boldsymbol{\theta}$ that it is associated with.

it might be more difficult to design models with a conjugate data distribution than for the CRP, because the data distribution now conditions on a varying number of parameters.

2.5.3 Hierarchical Processes

In the previous sections we saw that the Dirichlet process and the beta-Bernoulli process may use arbitrary base distributions. Most of the time, we did not specify the actual form of this base distribution or we used a normal-Wishart distribution when illustrating mixture models based on the Dirichlet process. However, because we may use arbitrary base distributions we may also use a draw from another process as the base distribution.

In short, the key idea of hierarchical processes is that the base distribution of a low-level process is a draw from a high-level process. For example, in a two-level hierarchical Dirichlet process [70, 71]

$$G_0 \sim \mathrm{DP}(\beta, H_0) \qquad \text{once} \qquad \text{high-level DP} \tag{2.106}$$

$$G_i \sim \mathrm{DP}(\alpha, G_0) \qquad \text{for } i = 1, \dots, n \qquad \text{low-level DP} \tag{2.107}$$

we have a high-level DP from which we once draw $G_0 = \{w_{0,j}, \theta_{0,j}\}_{j=1}^{\infty}$ which then will be reused as the base distribution for the low-level DP from which we draw several $G_i = \{w_{i,j}, \theta_{i,j}\}_{j=1}^{\infty}$. The draws G_i from the low-level DP are related, as they share the same set of atoms that also appear in G_0. However, the weights of the atoms may differ for each G_i. This is best illustrated by considering how a stick-breaking construction for a draw G_i works: The weights $w_{i,j}$ are only influenced by the parameter α of the low-level process, while the atoms $\theta_{i,j} \sim G_0$ are sampled independently from the base distribution

G_0. Because G_0 is a discrete distribution (an infinite set of weighted atoms), each atom that has been sampled during the stick-breaking construction of G_i is necessarily an atom that also appears in G_0. Further, an atom in G_0 may be sampled multiple times during the stick-breaking construction of G_i. Without changing the distribution G_i we could replace these multiple occurrences of an atom by a single occurrence and assign a new weight $\tilde{w}$ to it that corresponds to the sum of the weights of its multiple occurrences in G_i. If we write out G_i more formally as the sum of weighted Dirac functions then

$$G_i(\cdot) = \sum_{j=1}^{\infty} w_{i,j} \delta_{\theta_{i,j}}(\cdot) \tag{2.108}$$

$$= \sum_{h=1}^{\infty} \tilde{w}_{i,h} \delta_{\theta_{0,h}}(\cdot) \qquad \text{with } \tilde{w}_{i,h} = \sum_{k:\theta_{i,k}=\theta_{0,h}} w_{i,k} \text{ for each } h \tag{2.109}$$

Hence, each G_i can be considered as a re-weighting of the atoms of G_0. This can be useful for modeling related data sets, in which we expect to see the same clusters across different data sets but the clusters are allowed to have different weights for each data set.

There also exists a corresponding hierarchical Chinese restaurant process representation (called the “Chinese restaurant franchise” [3]) that arises when we integrate out the individual draws from this model. In this representation, we have a high-level CRP that corresponds to G_0 and several low-level CRPs that correspond to the draws G_i from the low-level process (see also Fig. 2.9). A customer enters a low-level CRP and is seated at one of the tables according to the usual CRP rules, whereby a new table is chosen with a probability proportional to α – this parameter stems from the low-level DP of Eq. (2.107). If a new table is chosen, we also need to draw a parameter for this newly occupied table. For this, we go to the high-level CRP, take a seat (according to the rules which we describe below), and use the table parameter of the corresponding high-level table as the new parameter for the low-level table. We may say, that the low-level table now references the high-level table (this corresponds to an arc in Fig. 2.9). Customers that arrive at an already occupied table of the low-level CRP would reuse the corresponding table parameter without being redirected to the high-level restaurant.

The seating rules for the high-level CRP are defined as follows. A new table is chosen

[3]Please note, that in the original paper [71] the hierarchical Chinese restaurant process is called the Chinese restaurant franchise and is presented by using a different metaphor in which the high-level CRP corresponds to a “menu” common to all (low-level) restaurants of a “franchise” from which the first customer of a new table chooses a “dish” (the table parameter). To avoid a confusion with the “dishes” of the Indian buffet metaphor, we described the menu here as a “high-level restaurant”.

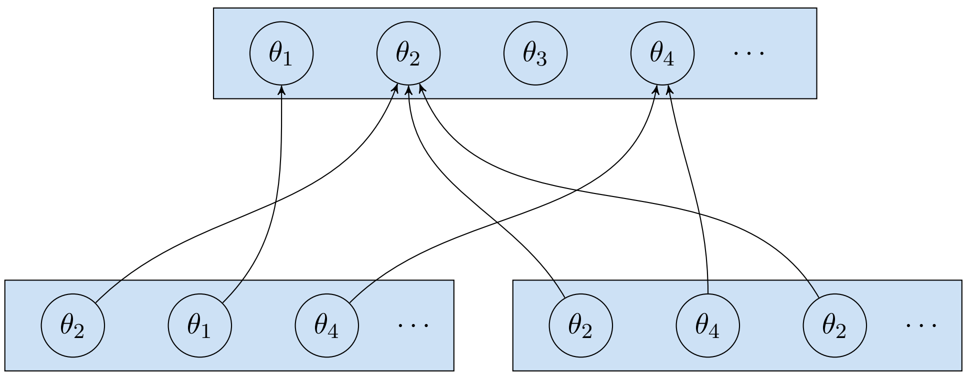


Figure 2.9: An illustration of a hierarchical Chinese restaurant process. The high-level CRP is shown on top and two low-level CRPs are depicted below. The table parameters of the low-level CRPs reference the table parameters in the high-level CRP. Data points (customers) would enter one of the low-level CRPs, while the customers of the high-level CRP correspond to the empty tables of the low-level CRPs. Hence, for the high-level CRP the number of customers at a table corresponds to the number of incoming arcs.

with a probability proportional to the parameter β, which stems from the definition of the high-level DP in Eq. (2.106). In this case, we need to a draw a parameter for this newly occupied high-level table from the base distribution H_0 of Eq. (2.106). An already occupied table in the high-level CRP is chosen with a probability proportional to the number of times it has been referenced by tables from *any* low-level restaurant. Hence, the customers in the high-level CRP can be considered to be the empty tables of the low-level CRPs for which we sampled a new table parameter. In Fig. 2.9, this count corresponds to the number of incoming arcs. The references are counted on a per-table basis – if several customers sit at the same low-level table, then this low-level table nevertheless counts as a single reference to the high-level table. Please note, that the counts of the high-level CRP thereby introduce the desired coupling between the different low-level CRPs – the distribution from which a new table parameter of a low-level restaurant is sampled, is also influenced by the table parameters of other low-level restaurants.

We also like to point out – without going into details – that similar hierarchical constructions also exists for the beta-Bernoulli process [73].

2.5.4 Nested Processes

In a hierarchical process, the base distribution is a *draw* from another process. In a nested process, the base distribution is itself a stochastic process. Hence, a two-level nested Dirich-

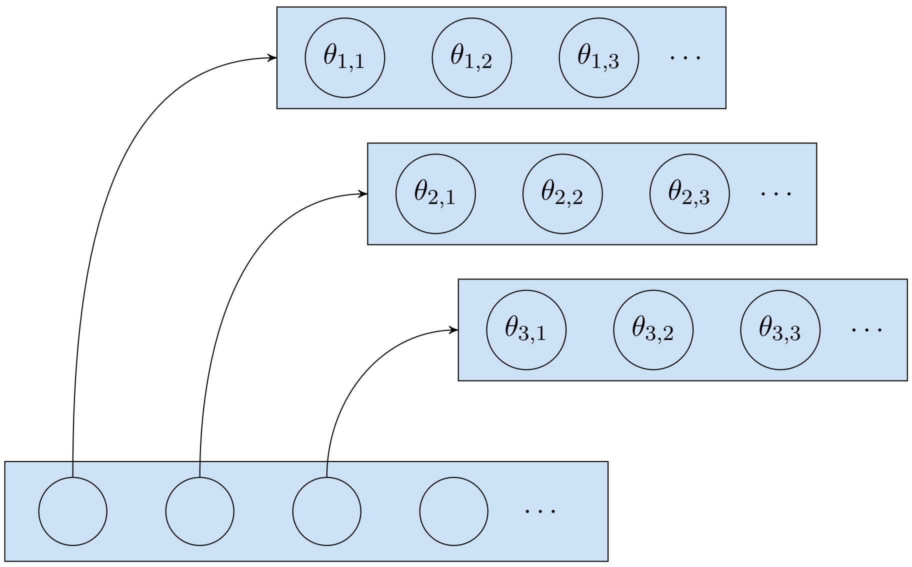


Figure 2.10: An illustration of a nested Chinese restaurant process. Each table parameter can be thought of as a reference to another CRP which is nested within this table. In contrast to the hierarchical CRP, the atoms $\theta_{i,j}$ are distinct (assuming a continuous base distribution) and are not shared among the different CRPs. Data points (customers) would enter the CRP on the lower left and would finally be seated at one of the tables of the referenced CRP on the upper right.

let process [57] can be defined as

$$G \sim \mathrm{DP}(\alpha, \mathrm{DP}(\beta, H_0)). \tag{2.110}$$

Again, the draw G can be represented as an infinite set $\{w_i, \theta_i\}_{i=1}^{\infty}$ of weighted atoms drawn from the base distribution. In this model, the base distribution is another DP. Hence, the atoms

$$\theta_i = G_i \sim \mathrm{DP}(\beta, H_0) \tag{2.111}$$

themselves are draws from the nested Dirichlet process and each G_i is again a set of weighted atoms, whereby these atoms are draws from H_0. If H_0 is continuous, then the atoms appearing in G_i and G_j (for some $i \neq j$) are distinct. Thus, in contrast to the hierarchical CRP, the atoms are not shared between the individual processes.

As noted in [7], the draws from nested Dirichlet process can be integrated out yielding a nested Chinese restaurant process, which is illustrated in Fig. 2.10. In this representation, each table parameter of the first CRP corresponds to a separate, nested CRP. One can imagine the table parameters as little notes with the address of another Chinese restaurant.

A customer arriving at a table of the first restaurant is transferred to the referenced second restaurant and is finally seated there. A Gibbs sampler for such a model needs to maintain and resample two association variables for each data point (or customer): one index for the table in the first restaurant and a second index for the table in the referenced restaurant. This model can also be extended to multiple levels in the obvious way.

Nested models therefore separate data points into distinct non-interacting groups and each group maintains its own set of atoms. In contrast, hierarchical models promote the sharing of information among low-level processes by maintaining a globally available set of atoms that can be referenced from the various low-level processes.

The Dirichlet process and the beta process can also be mixed in such models. For example, a beta process can be the base distribution in a Dirichlet process. In the usual metaphor, a customer arriving at a table in the Chinese restaurant would be transferred to an Indian restaurant with a large buffet, where she tastes different dishes. A draw from such a model would result in a set of atoms:

$$G \sim \mathrm{DP}(\alpha, \mathrm{BP}(c, \beta, H_0)) \tag{2.112}$$

$$G^{(i)} \sim G \tag{2.113}$$

$$\{\theta^{(i,1)}, \ldots, \theta^{(i,k)}\} \sim \mathrm{BeP}(G^{(i)}) \tag{2.114}$$

2.5.5 Gaussian Process Regression

The Dirichlet process and the beta process place a prior on *discrete* structures represented by an infinite set of weighted atoms. In Gaussian process regression, we place a prior on *continuous* functions. This is useful for regression, in which we are given a set of points and are interested in the posterior distribution over functions that are consistent with our observations. Based on this posterior distribution we can evaluate the predictive likelihood for observing a point at a new location given the previously seen data points.

More formally, in Gaussian process regression [54], we are given a set $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$ of noisy observations $y_i = f(\mathbf{x}_i) + \epsilon$ of an unknown function $f : \mathbb{R}^n \to \mathbb{R}$ evaluated at input locations $\mathbf{x}_i$. The additive noise ϵ is assumed to be Gaussian $\epsilon \sim \mathcal{N}(0, \sigma_n^2)$ with zero mean and variance σ_n^2. The objective is to estimate the function value $f(\mathbf{x}^*)$ at an arbitrary location $\mathbf{x}^*$ given the data $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$.

Gaussian processes (GPs) are a nonparametric approach to the regression problem, that is, we do not have to specify a parametric form of the unknown function f and fit the function parameters to the data. Instead, a GP can be considered as a distribution over functions. A GP is specified by a mean function $m(\cdot)$ and covariance function $k(\cdot,\cdot)$

$$m(\mathbf{x}) = E[f(\mathbf{x})] \tag{2.115}$$

$$k(\mathbf{x},\mathbf{x}') = E[f(\mathbf{x}) - m(\mathbf{x})]E[f(\mathbf{x}') - m(\mathbf{x}')] \tag{2.116}$$

that imply a prior distribution over functions before seeing any data and, informally speaking, capture our prior knowledge about the nature of the unknown function f. For the sake of simplicity, we will assume a zero mean function for the remainder of this section. A common choice for the covariance function is the squared exponential

$$k(\mathbf{x},\mathbf{x}') = \sigma_f \exp\left(\frac{\|\mathbf{x}-\mathbf{x}'\|^2}{2l^2}\right). \tag{2.117}$$

If we sample m function values $\mathbf{y} = (y_1,\ldots,y_m)^T$ at locations $X = \{\mathbf{x}_i\}_{i=1}^m$ from this prior over functions, then they will be distributed according to a multivariate Gaussian distribution

$$\mathbf{y} \sim \mathcal{N}(0, K(X,X) + \sigma_n^2\mathbf{I}) \tag{2.118}$$

where $K(X,X)$ is a covariance matrix with entries $k_{ij} = k(\mathbf{x}_i,\mathbf{x}_j)$. Accordingly, the noisy observations $\mathbf{y} = (y_1,\ldots,y_n)^T$ and the function value y^* at a query point $\mathbf{x}^*$ will be jointly Gaussian distributed according to

$$\begin{bmatrix}\mathbf{y}\\ y^*\end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix}K(X,X)+\mathbf{I}\sigma_n^2 & K(X,\mathbf{x}^*)\\ K(\mathbf{x}^*,X) & k(\mathbf{x}^*,\mathbf{x}^*)\end{bmatrix}\right) \tag{2.119}$$

We can now condition this distribution on the data $\mathcal{D} = \{(\mathbf{x}_i,y_i)\}_{i=1}^n$ and obtain the predictive distribution for the query point $\mathbf{x}^*$ as a Gaussian $\mathcal{N}(\mu_{\mathbf{x}^*},\sigma_{\mathbf{x}^*}^2)$ with mean and covariance

$$\mu_{\mathbf{x}^*} = K(X,\mathbf{x}^*)^T\left(K+\sigma_n^2\mathbf{I}\right)^{-1}\mathbf{y} \tag{2.120}$$

$$\sigma_{\mathbf{x}^*}^2 = k(\mathbf{x}^*,\mathbf{x}^*) - K(X,\mathbf{x}^*)^T\left(K+\sigma_n^2\mathbf{I}\right)^{-1}K(X,\mathbf{x}^*). \tag{2.121}$$

If we are interested in predicting several query points, then the predictive distribution will be a multivariate Gaussian distribution which thereby also models to covariance between the query points.

CHAPTER 3

RFID-based Object Localization and Self-Localization

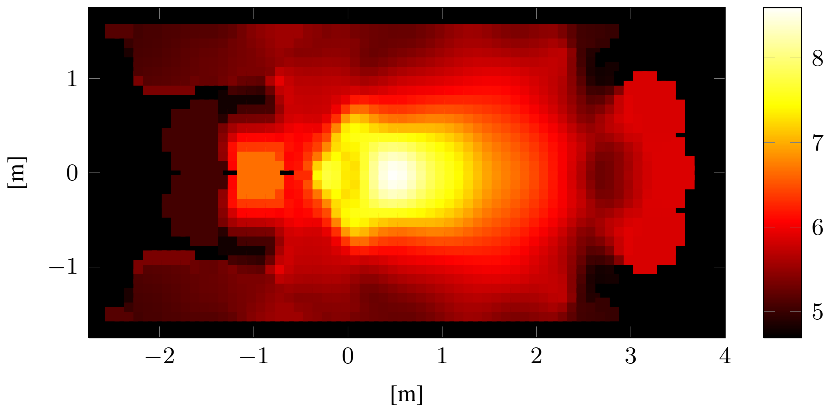
In recent years, there has been an increasing interest within the robotics community in investigating whether Radio Frequency Identification (RFID) technology can be utilized to solve localization and mapping problems in the context of mobile robots. We present a novel sensor model which can be utilized for localizing RFID tags and for tracking a mobile robot moving through an RFID-equipped environment. The proposed probabilistic sensor model characterizes the received signal strength indication (RSSI) information as well as the tag detection events to achieve a higher modeling accuracy compared to state-of-the-art models which deal with one of these aspects only. We furthermore propose a method that is able to bootstrap such a sensor model in a fully unsupervised fashion. Real-world experiments demonstrate the effectiveness of our approach also in comparison to existing techniques.

Radio Frequency Identification (RFID) technology has become popular in areas such as supply chain management and inventory control, primarily because information can be attached to real-world products cheaply and can be retrieved without requiring physical contact. Recently, also robotics researchers have started to explore potential applications of the technology, focusing on the tasks of localization, mapping, and activity recognition. An RFID system consists of one or several RFID antennas and tags distributed in the environment. The antenna sends out electromagnetic waves and the passive RFID tags, consisting of a chip and a small antenna, use either load modulation or backscattering to send back their unique ID to the receiver. Ultra-high frequency (UHF) systems, one of which is also used in this work, has a reading range of up to a few meters.

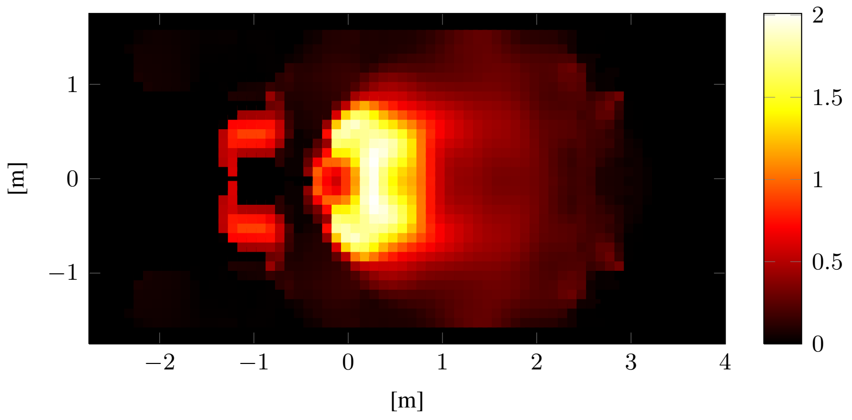
According to scenarios envisioned for the near future, virtually every retail product could be equipped with an RFID tag. In addition to the *semantic* information about a given environment or situation [51, 40], such a setup also provides a rich source of *spatial* information, which can be utilized (a) to infer the location of the tags within the environment, (b) to localize the sensor relative to them, or (c) to solve both tasks jointly. A major precondition for solving all these tasks is the availability of an accurate sensor model $p(\mathbf{z} \mid \mathbf{x})$ that characterizes the relationship between locations $\mathbf{x}$ and measurements $\mathbf{z}$. The contribution of the work presented in this chapter is two-fold: First, we present a novel sensor model that utilizes the received signal strength indication (RSSI) as well as tag detection events to achieve superior accuracy compared to state-of-the-art models that cover one of the two aspects only (see Fig. 3.1 for an illustration). We use this sensor model for localizing RFID tags and for tracking a mobile sensor platform, a shopping cart, equipped with two RFID antennas. As a second contribution, we describe how our sensor model can be learned in an fully unsupervised fashion. We also compare our model with a sensor model that has been shown to be effective for WiFi localization and we point out how this model can be further improved in the context of RFID localization and present experimental results. Real-world experiments in an office environment and a supermarket demonstrate that our system is able to robustly estimate the position of the tags and to track a shopping cart moving through an RFID-equipped supermarket.

3.1 The Sensor Model

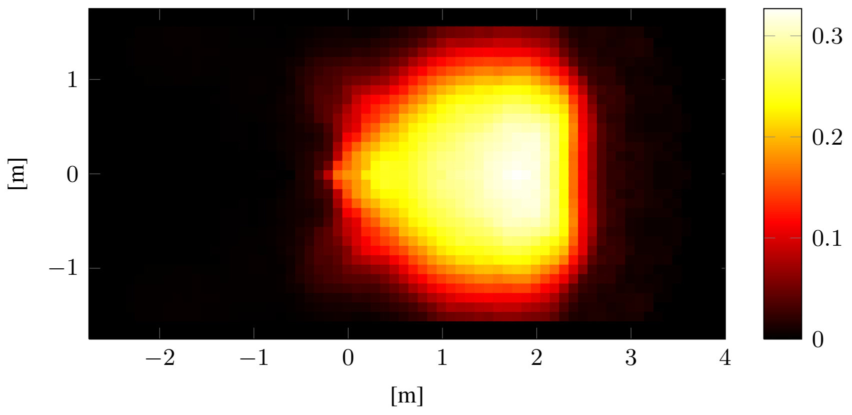
The techniques for localizing RFID tags as well as for tracking a mobile antenna both rely on a sensor model $p(\mathbf{z} \mid \mathbf{x}, \boldsymbol{\ell}_g)$ which specifies the likelihood of obtaining a measurement $\mathbf{z}$ given the pose $\mathbf{x} = (x, y, \theta)$ of the antenna and the location $\boldsymbol{\ell}_g = (x_g, y_g)$ of the detected tag with unique ID g. In our case, an observation $\mathbf{z} = (g, s)$ carries two pieces of information, namely that we have detected the tag g in the first place, and secondly that we received its signal with a signal strength s. Indeed, the event of detecting a tag is informative in itself, and this fact forms the basis for many of the previously proposed probabilistic sensor models for tag localization [26, 15, 43, 80]. Note also that the other class of existing sensor models, which consider signal strength only, implicitly condition on the tag detection event



(a) Mean of the logarithmic signal strength.



(b) Standard deviation of the logarithmic signal strength.



(c) Tag detection probability.

Figure 3.1: The proposed sensor model combines information about the expected received signal strength, (a) and (b), and about the probability of detecting a tag (c).

since a signal strength measurement can be obtained in this way only. We make the distinction between both sources of information explicit by denoting with d the binary variable that encodes the detection of a certain tag. Hence, the sensor model can be formalized as

$$\begin{aligned} p(\mathbf{z} \mid \mathbf{x}, \boldsymbol{\ell}_g) &= p(s, d \mid \mathbf{x}, \boldsymbol{\ell}_g) \\ &= p(s \mid d, \mathbf{x}, \boldsymbol{\ell}_g) p(d \mid \mathbf{x}, \boldsymbol{\ell}_g). \end{aligned} \tag{3.1}$$

In the most general form, these two conditional distributions are intractable to learn in practice since, for example, $p(d \mid \mathbf{x}, \boldsymbol{\ell}_g)$ specifies a tag detection probability for every possible combination of antenna pose $\mathbf{x}$ and tag location $\boldsymbol{\ell}_g$. Therefore, we make the common assumption that only the *relative* location $\Delta(\mathbf{x}, \boldsymbol{\ell}_g)$ of a tag with respect to the antenna is relevant (see [26, 15, 43, 80]). This assumption certainly is a strong one, since the propagation of an RFID signal is also influenced by *location-dependent* factors, such as the materials the tags are attached to, the orientation of the tags relative to the antenna, or obstacles that reflect or absorb electro magnetic waves. The gain in efficiency, however, is large in comparison to other simplifications that could be made. As we will show in the experimental evaluation, the accuracy of a *location-dependent* sensor model for RFID tags is slightly higher than of our model, but that gain comes at a high computational cost already for small environments.

Committing ourselves to *sensor-centric* sensor modeling, which considers relative tag positions only as outlined above, we get

$$p(\mathbf{z} \mid \mathbf{x}, \boldsymbol{\ell}_g) = p(s \mid d, \Delta(\mathbf{x}, \boldsymbol{\ell}_g)) p(d \mid \Delta(\mathbf{x}, \boldsymbol{\ell}_g)) \,. \tag{3.2}$$

In words, this models the likelihood of an observation as the likelihood of receiving signal strength s at position $\Delta(\mathbf{x}, \boldsymbol{\ell}_g)$ relative to the antenna multiplied by the probability of detecting a tag at this relative position.

Location-dependent sensor models, that characterize the distribution of signal strength relative to the environment rather than to the sensor (see [64, 62, 19]), can be understood as a different approximation that stays more faithful to the true signal strength distribution. Instead of conditioning the signal strength on the relative tag position, they learn a separate signal strength distribution $p_g(s \mid \mathbf{x})$ for each tag individually, which is conditioned only on the antenna location. The resulting signal strength *maps* implicitly contain all environment-specific factors. On the downside, they do not model the location of the tag explicitly and, thus, cannot be used to estimate the location of the tags directly.

3.2 Learning the Model from Data

We first describe how the components of Eq. (3.2) can be learned in a semi-autonomous way and then extend this procedure to an unsupervised bootstrapping method.

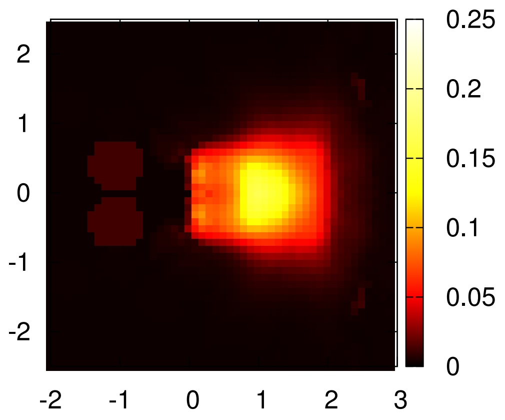
3.2.1 Semi-Autonomous Learning

Vorst *et al.* [80] proposed a method for learning a tag detection sensor model in a semi-autonomous fashion, which we will adopt and extend towards also learning the signal strength distribution. Building on this, we show how to learn both models in an fully autonomous way. The semi-autonomous way of learning a tag detection model is to assume a list of tag positions given as well as the trajectory of a mobile antenna moving through the environment. At every tag detection event, we transform the tag positions into the antenna's local coordinate system and register the tag detection at the tags relative position as a positive event while registering the non-detected tags as negative events. We then discretize the space relative to the antenna according to a two-dimensional grid and count for every grid cell (x, y) the positive events $n^+_{x,y}$ and the negative events $n^-_{x,y}$. Given these counts, the tag detection probability is then defined as $p_{x,y} = n^+_{x,y} / \left(n^+_{x,y} + n^-_{x,y}\right)$. Likewise, we maintain a second grid that contains statistics about the average received logarithmic signal strength $\mu_{x,y}$ and the empirical variance $\sigma_{x,y}$ for each grid cell. Under the assumption that the logarithmic signal strengths within each grid cell are normally distributed, we can estimate the likelihood of an observation $\mathbf{z} = (g, s)$ at the antenna relative position $\Delta(\mathbf{x}_t, \boldsymbol{\ell}_g) = (x, y)$ as

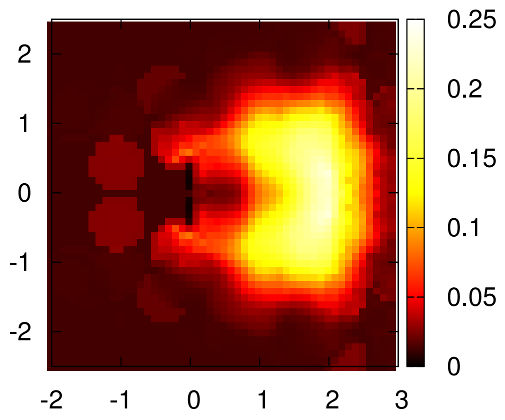
$$\begin{aligned} p(\mathbf{z} \mid \mathbf{x}, \boldsymbol{\ell}_g) &= p(s \mid d, \Delta(\mathbf{x}, \boldsymbol{\ell}_g)) p(d \mid \Delta(\mathbf{x}, \boldsymbol{\ell}_g)) \\ &\propto \frac{1}{\sigma_{x,y}\sqrt{2\pi}} \exp\left(-\frac{(\log(s) - \mu_{x,y})^2}{2\sigma^2_{x,y}}\right) p_{x,y} \,. \end{aligned} \tag{3.3}$$

3.2.2 Bootstrapping the Sensor Model

So far, learning the sensor models required knowledge about the true tag positions, which might be tedious or impossible to acquire in practice. We can sidestep this, by bootstrapping the sensor models. We start with a basic, yet plausible tag detection model similar to the one proposed by Hähnel *et al.* [26] and iterate the following steps: (a) Use the current model to estimate the tag locations (as described in the next section) and (b) learn a new sensor model based on the estimated tag locations (as described above).



(a) RSSI=log(2000)



(b) RSSI=log(500)

Figure 3.2: Two instances of the inverse sensor model that describes the likelihood of relative offsets between antenna and tag given a certain level of received signal strength (RSSI). We plot (a) RSSI=log(2000) and (b) RSSI=log(500). The antenna is located at $(0, 0)$ and oriented towards the positive x-axis.

In the experimental section, we present results that indicate that alternating tag location estimation and sensor model learning converges in terms of tag location error and the similarity of the bootstrapped sensor model to a model learned in a semi-autonomous way. Fig. 3.1 shows an example for a sensor model learned this way in an office environment. Fig. 3.2 visualizes two instances of the corresponding *inverse* sensor model, that is, the likelihood of relative poses *given* a certain level of received signal strength. This model has been calculated analytically from the three components of the sensor model depicted in Fig. 3.1. The improvement of the estimated tag locations during the bootstrapping procedure is illustrated in Fig. 3.3.

Based on this bootstrapping procedure, we can learn both the sensor model and the tag locations in a fully unsupervised fashion. This greatly simplifies sensor modeling in practice, compared to the manual acquisition of calibration data. In contrast to the semi-autonomous method, it does not require knowledge about the true tag positions.

3.3 Mapping Tags from Known Sensor Poses

For localizing RFID tags, we move a mobile antenna through the environment and integrate the measurements iteratively so that the estimates of tag locations improve gradually over time. We assume that the antenna is localized, e.g., applying laser-based FastSLAM [25],

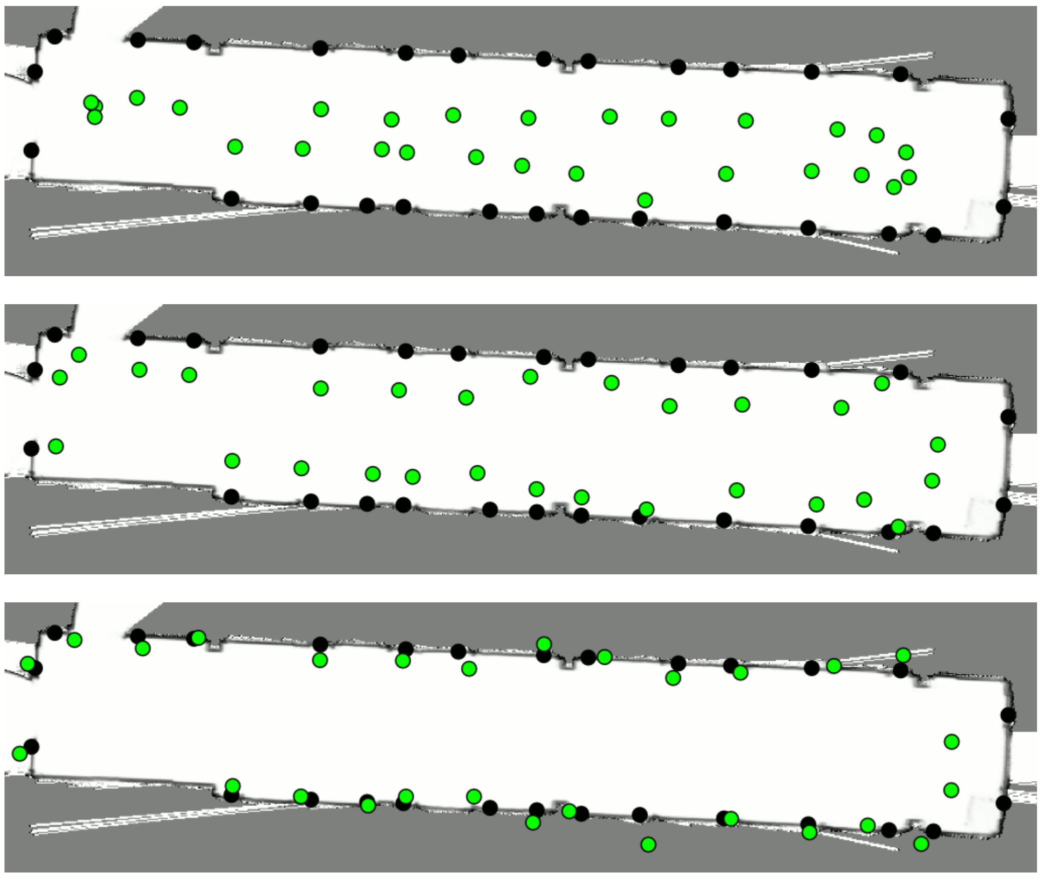


Figure 3.3: Ground truth tag locations (black) and tag locations estimated (green) with the initial sensor model (top), and with the learned models at bootstrapping iterations 2 (middle) and 25 (bottom).

so that we have accurate estimates of the positions at which observations have been made.

Formally, we are given a sequence of tag readings $\mathbf{z}_{1:t} = \{(g_i, s_i)\}_{i=1}^{t}$ denoting the unique ID g of the detected tag and the received signal strength s, as well as a sequence of antenna poses $\mathbf{x}_{1:t} = \{(x_i, y_i, \theta_i)\}_{i=1}^{t}$, denoting the antenna's position and orientation at which these observations have been made. We are interested in the posterior $p(\boldsymbol{\ell}_g \mid \mathbf{x}_{1:t}, \mathbf{z}_{1:t})$ of the tag location $\boldsymbol{\ell}_g = (x_g, y_g)$ given the information up to time t. Using Bayes' rule and assuming independence between measurements, we get the recursive update formula

$$p(\boldsymbol{\ell}_g \mid \mathbf{x}_{1:t}, \mathbf{z}_{1:t}) = \eta p(\mathbf{z}_t \mid \Delta(\mathbf{x}_t, \boldsymbol{\ell}_g)) p(\boldsymbol{\ell}_g \mid \mathbf{x}_{1:t-1}, \mathbf{z}_{1:t-1}), \tag{3.4}$$

where $p(\mathbf{z}_t \mid \Delta(\mathbf{x}_t, \boldsymbol{\ell}_g))$ is the sensor model described in the previous section and η is a normalization factor (see also [26]).

To estimate this posterior sequentially as new data arrives, we apply a particle filter for each tag and use its unique ID g for data association. Each filter is initialized with a uniform particle distribution $\{(w_g^{(i)}, \boldsymbol{\ell}_g^{(i)})\}_i$, where $w_g^{(i)}$ denotes the weight of a particle and $\boldsymbol{\ell}_g^{(i)}$ its location. The initial particle distribution is bounded by the maximum reading range of the antenna and centered around the antenna's position during the first encounter of the tag [visualized in Fig. 3.4 (a)]. We resample whenever the so-called number of effective particles

$$n_{\text{eff}} = \frac{1}{\sum_i (w^{(i)})^2} \tag{3.5}$$

falls below a threshold κ, which we set to half the number of particles of the filter. In the resampling step, we follow Liu and West [42], and disturb the individual particle locations by resampling them from a normal distribution with the following parametrization

$$\boldsymbol{\ell}_g^{(i)} \sim \mathcal{N}(\boldsymbol{\mu}_{\text{LW}}, \boldsymbol{\Sigma}_{\text{LW}}) \tag{3.6}$$

$$\boldsymbol{\mu}_{\text{LW}} = a\boldsymbol{\ell}_g^{(i)} + (1-a)\boldsymbol{\mu}_g \tag{3.7}$$

$$\boldsymbol{\Sigma}_{\text{LW}} = h^2 \boldsymbol{\Sigma}_g \tag{3.8}$$

Here, $\boldsymbol{\mu}_g$ and $\boldsymbol{\Sigma}_g$ are the mean and covariance matrix of the particle set of tag g and the two parameters a and h^2 are defined as

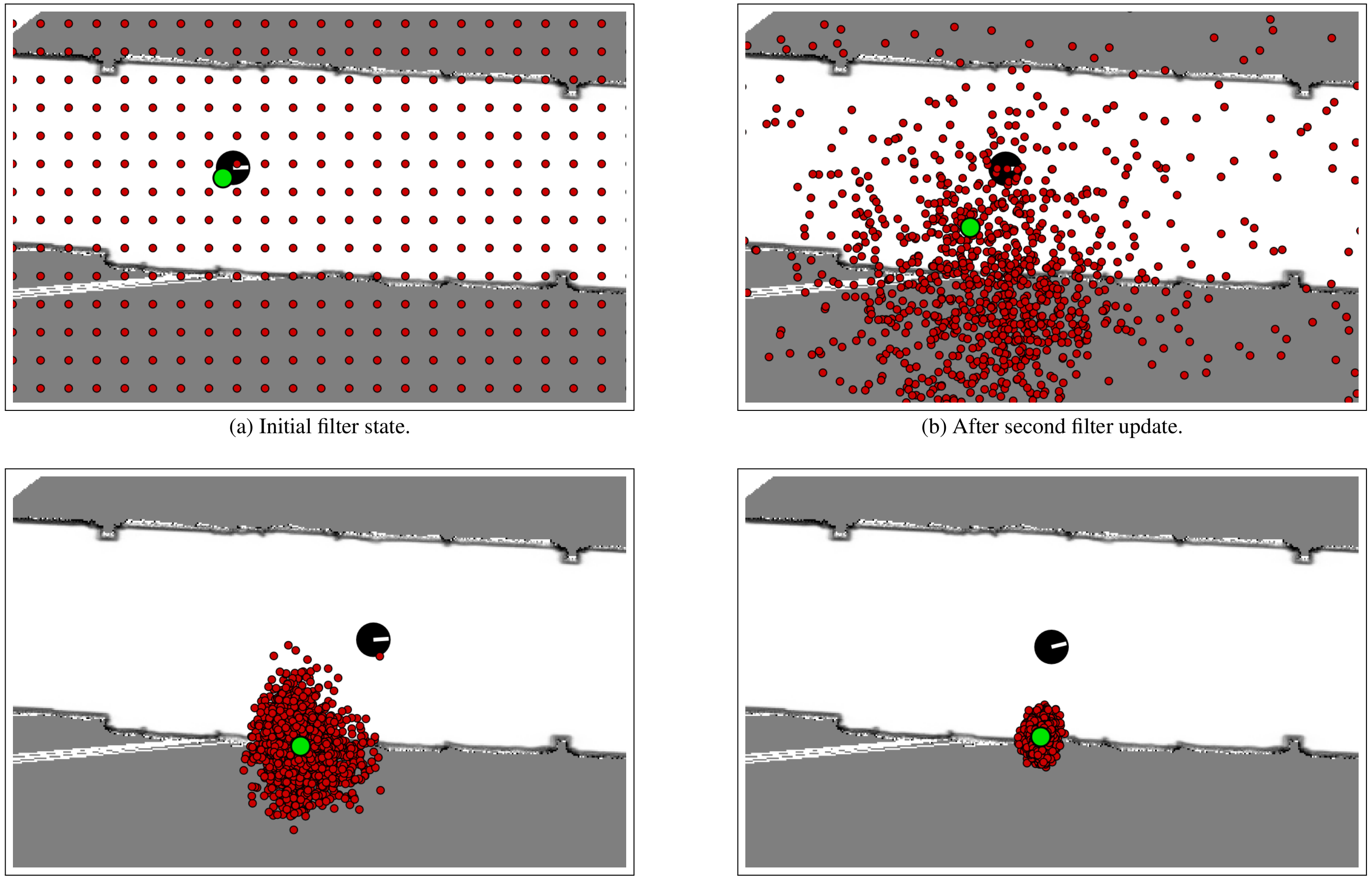
$$a = \frac{3\gamma - 1}{2\gamma} \tag{3.9}$$

$$h^2 = 1 - a^2 \tag{3.10}$$

and only depend on a discount factor γ, which we set to 0.95. This procedure causes the particle-based estimates of the tag locations to converge to the true locations even for crudely initialized estimates within a few filter iterations, as can be seen in Fig. 3.4.

3.4 Localizing a Mobile Agent

Given that we know the locations of the RFID tags, we can use the very same sensor model to track a mobile agent equipped with an RFID antenna. We apply Monte Carlo Localization (MCL) [11], which utilizes a particle filter to maintain the posterior over the



(a) Initial filter state.

(b) After second filter update.

(c) After 5th filter update.

(d) After 30th filter update.

Figure 3.4: Localizing RFID tags in an office corridor: The green circle indicates the estimated tag location and the black circle the pose of the shopping cart.

agent's location

$$\mathrm{bel}(\mathbf{x}_{t+1}) \propto p(\mathbf{z}_{t+1} \mid \Delta(\mathbf{x}_{t+1}, \boldsymbol{\ell}_g)) \int_{\mathbf{x}_t} p(\mathbf{x}_{t+1} \mid \mathbf{x}_t)\, \mathrm{bel}(\mathbf{x}_t)\, d\mathbf{x}_t\,. \tag{3.11}$$

While the sensor model $p(\mathbf{z} \mid \Delta(\mathbf{x}, \boldsymbol{\ell}_g))$ remains the same, the crucial part here is the motion model $p(\mathbf{x}_{t+1} \mid \mathbf{x}_t)$, from which we sample the next particle distribution. As we are tracking a shopping cart, which does not provide odometry information, we used a velocity based motion model that tries to capture the typical motion patterns of people pushing the cart. For this, each particle $\mathbf{s}^{(i)}$ is constrained in a seven-dimensional space

$$\mathbf{s}^{(i)} = (\mathbf{x}, \mathbf{v}, m) = (x, y, \theta, v_x, v_y, v_\theta, m)\,, \tag{3.12}$$

that consists of the current pose $\mathbf{x} = (x, y, \theta)$ of the cart, its velocity $\mathbf{v} = (v_x, v_y, v_\theta)$, and a motion state m. The velocity is parameterized by the translational velocity v_x in direction of the cart, a lateral drift velocity v_y, and a rotational velocity v_θ. The discrete motion state m can assume one of the following seven values: *standing*, *moving forwards (backwards)*, *turning left forwards (backwards)*, or *turning right forwards (backwards)*.

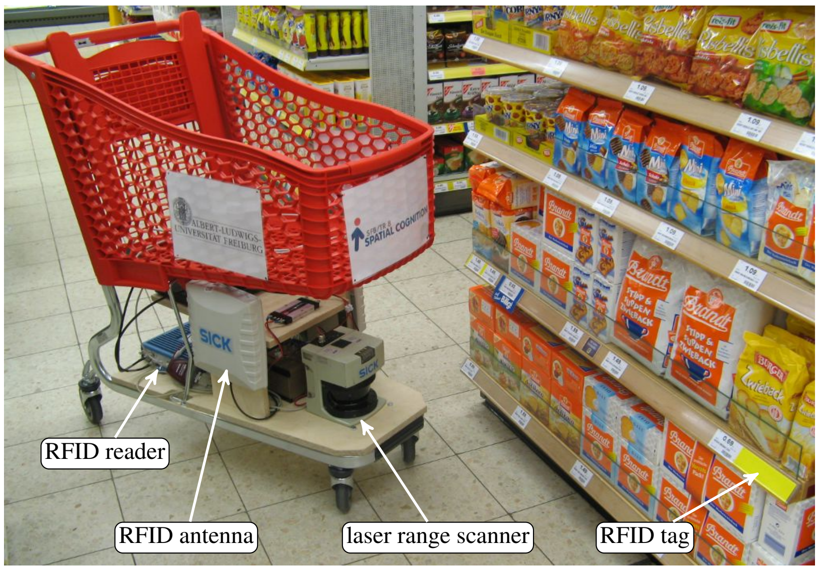
The transition from one particle state $\mathbf{s}_t^{(i)}$ to the state $\mathbf{s}_{t+1}^{(i)}$ at the next point in time is modeled as follows: first, we sample a new motion state m_{t+1} according to a state transition probability $p(m_{t+1} \mid m_t)$. If the motion state changed, we sample new velocities $\mathbf{v}_{t+1}$ from three motion state-specific normal distributions

$$v_x \sim \mathcal{N}(\mu_{m,x}, \sigma^2_{m,x}) \tag{3.13}$$

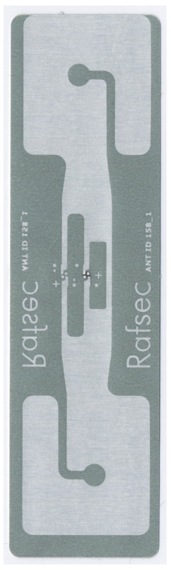
$$v_y \sim \mathcal{N}(\mu_{m,y}, \sigma^2_{m,y}) \tag{3.14}$$

$$v_\theta \sim \mathcal{N}(\mu_{m,\theta}, \sigma^2_{m,\theta}) \tag{3.15}$$

that capture the velocity distributions of the particular motion state. If the sampled motion state is the same as in the time step before, we do not sample new velocities, but rather keep the velocities of the previous point in time. Once the new velocities are determined, we deterministically compute the new pose of a particle based on the velocities and the time passed during one filter update step. As during tag localization, we resample whenever the number of effective particles is less than half of the particle set size. To account for physical constraints imposed by walls and other obstacles, we set the weight of a particle close to zero, whenever it enters an occupancy grid cell that is likely to be occupied. The transition probabilities $p(m_{t+1} \mid m_t)$ and the parameters of the velocity distributions ($\mu_{m,x}$, $\sigma^2_{m,x}$,



(a)



(b)

Figure 3.5: (a) We equipped a shopping cart with an RFID reader and a laser range scanner and deployed about 350 passive UHF tags in a supermarket. (b) DogBone RFID tag by UPM Raflatac.

etc.) were learned from recorded trajectories with hand-labeled motion states.

3.5 Experimental Evaluation

To evaluate our approach, we equipped a shopping cart with a SICK RFI 641 UHF RFID reader with two antennas mounted perpendicular to each side of the cart (see Fig. 3.5). The reader also reports which antenna detected the tag, and we know their positions relative to the center of the cart. As the antennas are identical in construction we assume the same sensor model for both. The reader is configured to run in continuous mode, reporting a tag as soon as it is detected. The typical tag detection rate of the system is about 10 Hz. We used passive UHF tags ("DogBone" by UPM Raflatac). In order to acquire a ground truth trajectory and an occupancy grid of the environment we additionally equipped the cart with a SICK LMS 291 laser scanner and processed the data with the GMapping algorithm [25], which is an efficient laser-based realization of the FastSLAM approach. In the remainder of this section we present experimental results about the tag localization approach and the localization of a mobile agent.

3.5.1 Localizing the RFID Tags

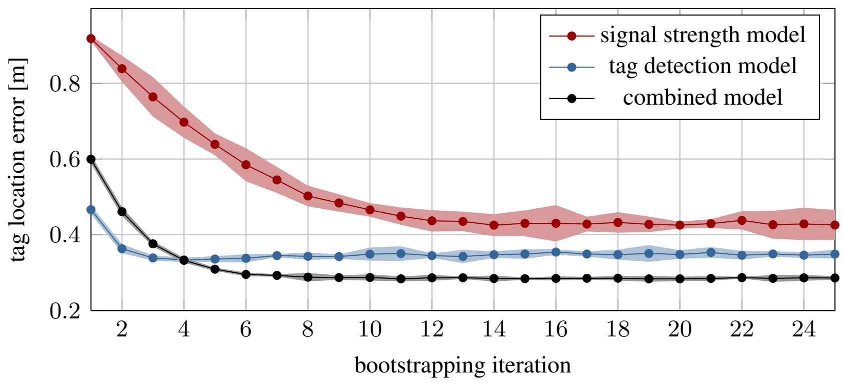
We distributed 28 RFID tags in an office corridor as depicted by the black circles in Fig. 3.3 on page 59. Neighboring tags had an average distance of about two meters. We knew the true locations of the tags and therefore could evaluate the accuracy of tag localization quantitatively. We bootstrapped the sensor model by moving the shopping cart up and down the corridor several times – performing 360° turns at several locations. This took about 4 minutes and resulted in roughly 4 100 tag detections. Fig. 3.6 (a) shows the evolution of the average error of the estimated tag locations after each iteration of the bootstrapping process. As can be seen, the error converges to a final value of about 29 cm. We also give the estimation results for the signal strength-based model alone (red line) and the tag detection-based model alone (blue line) evaluated on the same trajectory. Our proposed combined sensor model (black line) is significantly better than either the signal strength-based model or the tag detection-based one alone.

If we learn the sensor model semi-autonomously based on the true tag locations, we achieve a localization accuracy of about 27 cm instead of 29 cm. This indicates, that the bootstrapping process yields a sensor model that is comparable to a semi-autonomously learned sensor model. This was also confirmed by visually comparing the individual components of the two models. To confirm this finding quantitatively, we calculated the average symmetric Kullback-Leibler divergence between all grid cells of the two models after each bootstrapping iteration. The results illustrated in Figs. 3.6 (b) and (c) show that the bootstrapping process also converges in terms of model similarity to a semi-autonomously learned model.

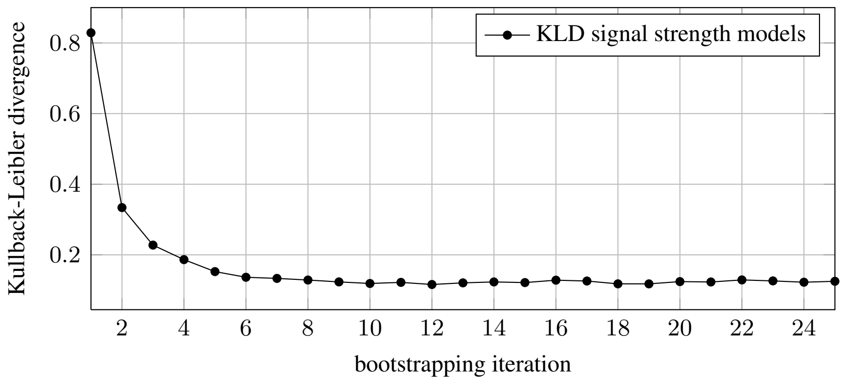
3.5.2 Localizing a Mobile Agent

We distributed about 350 tags along the shelves in a supermarket at an average distance of approximately one meter. Then we bootstrapped the sensor models and the tag positions by using data from six log files, which we collected by moving the shopping cart through the environment. The log files contained 34 200 tag detections and lasted about 74 minutes in total. We used the localization technique described above and defined as the localization result the trajectory of the most likely particle. An example of an estimated trajectory and its corresponding ground truth trajectory is depicted in Fig. 3.7.

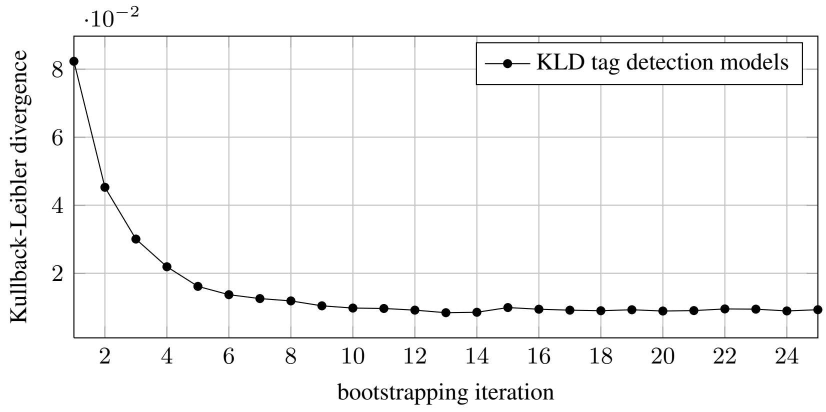
To evaluate the accuracy of the localization technique quantitatively, we localized the agent on seven different log files which lasted 24 minutes in total and contained 13 400 tag detections. We repeatedly localized the cart for each log file ten times and averaged the



(a) Average tag localization error.



(b) Divergence of the signal strength models.



(c) Divergence of the tag detection models.

Figure 3.6: The bootstrapping process: (a) The process converges in terms of the average tag location error. The error bars depict the 2σ confidence interval. (b) and (c) show that the bootstrapping procedure also converges in terms of sensor model similarity to a semi-autonomously learned model.

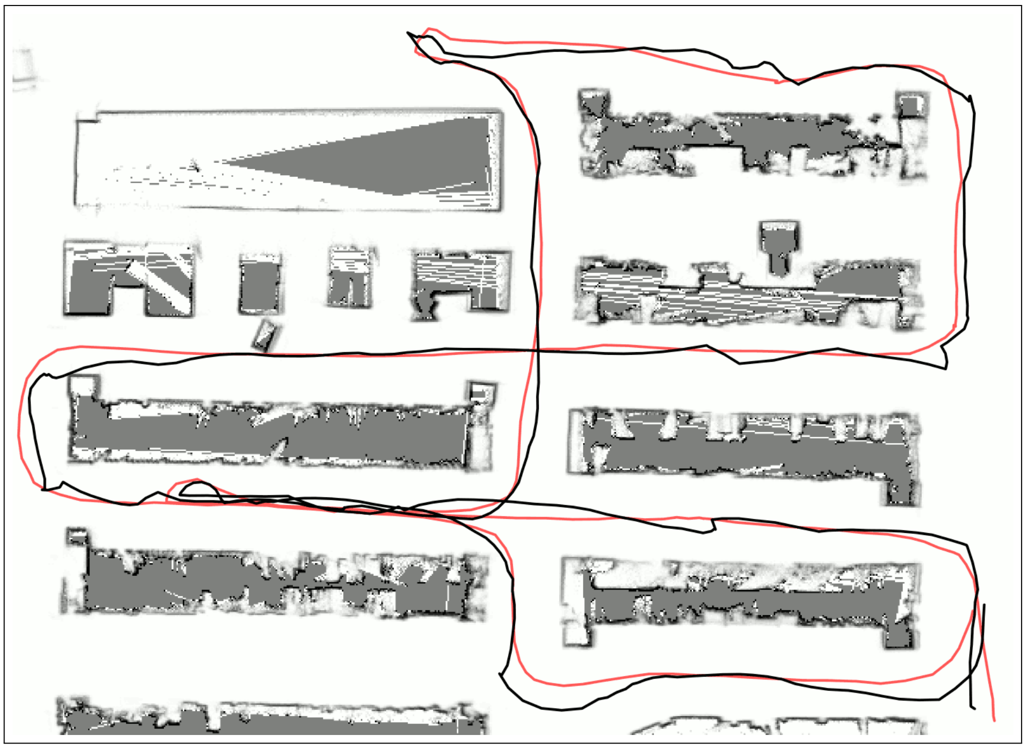
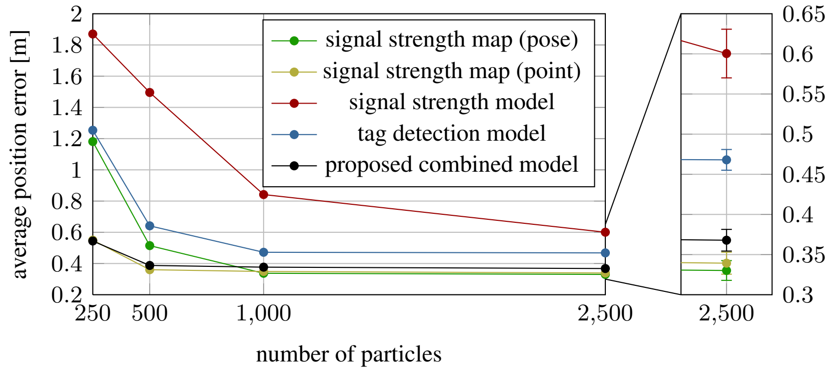


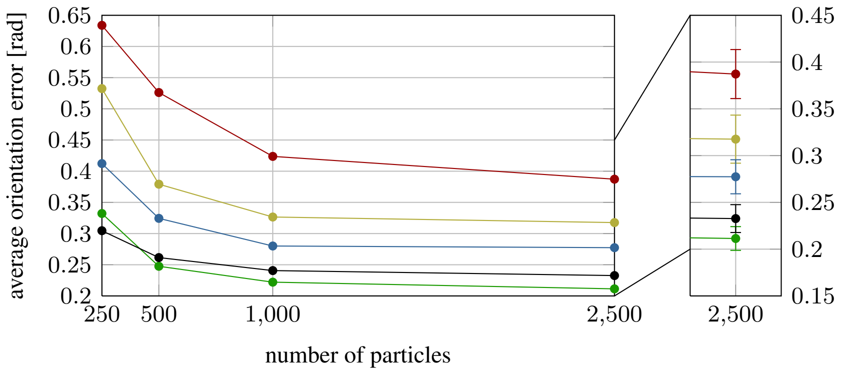
Figure 3.7: Comparison of the estimated trajectory (black) and the ground truth trajectory (red) on one of the supermarket log files.

measured error values. The error was quantified in terms of the average position error and the average orientation error. The results are given in Fig. 3.8.

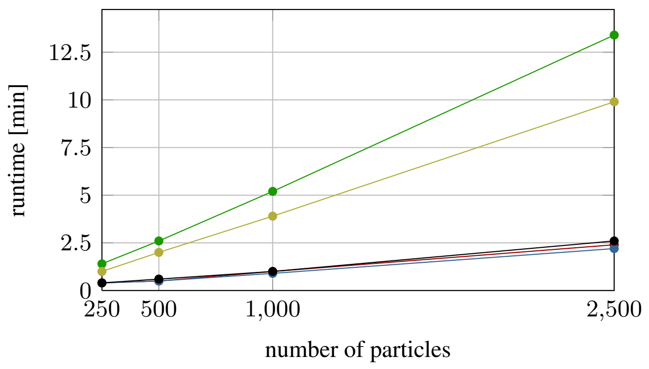
Using only the tag detection model can be considered equivalent to the approach by Vorst *et al.* [80]. For further comparison, we implemented a model similar to the one presented by Ferris *et al.* [19], which used Gaussian process regression [54] to model the signal strength distributions of WLAN access points in 2D space. We used Gaussian process regression for modeling the signal strength of the RFID tags in the supermarket. We observed that the estimation of the cart's orientation can be significantly improved [Fig. 3.8 (b)], if the antenna's orientation is taken into account, and hence the signal strength is mapped in pose space rather than in 2D. The proposed combined sensor model outperforms both of its components – the tag detection model and the signal strength model – and performs comparable to the signal strength map. Both methods can be executed online for 2 500 particles, but the proposed model needs only 2.6 minutes on average to process all log files, while the signal strength map, which uses Gaussian process regression instead of a grid, requires



(a) Average position error.



(b) Average orientation error.



(c) Runtime.

Figure 3.8: Evaluation of several sensor models for RFID-based localization: the proposed combined model, and its two components alone. For comparison, we show the results of a signal strength map, similar to the model by Ferris *et al.* [19], mapping the signal strength either in pose space ("pose") or in 2D ("point").

13.4 minutes, as can be seen in Fig. 3.8 (c).

In both tasks – localization and mapping – the signal strength model alone was consistently less accurate than the tag detection model. This is an unexpected finding, since the received signal strength should intuitively be more informative than a simple detection event. Close inspection of the recorded data reveals, however, that the relationship between signal-strength and sensor location is more noisy than it is the case for tag detections. Thus, the particle filter as a sophisticated way of integrating information over time, is able to recover the (real-valued) pose information accurately from the stream of (binary) tag detection events.

3.6 Future Work

There are several directions for future work. Our results showed that considering signal strength information along with tag detection events lead to an improved sensor centric model. Therefore, it would be interesting to see if the accuracy of signal strength maps could be improved in just the same way by combining them with "tag detection maps". Another direction would be to extend the model to a 3D sensor model. Moreover, the assumption made that the tag map remains static could be alleviated – a technique for mapping "nomadic" tags (static tags which change locations from time to time) was presented in [43].

3.7 Related Work

There is a variety of approaches to RFID-based localization, which can be characterized by the type of sensor information used as well as by the general approach to modeling this information. Some of the earlier systems only provided information about the ID of the detected tag while later systems also provide information about the received signal strength. Hence, some localization techniques utilize sensor models that are based only on tag detection events [26, 37, 60, 15, 43]. For some RFID readers that do not provide RSSI (signal strength) directly, this information can be emulated by means of different attenuation levels or power levels of the antenna [48, 1, 36]. Sensor models for localizing tags are usually designed to be sensor centric [26, 15, 43, 1, 12]. A different design, that can be used for localizing a mobile robot, is to lay out the sensor model relative to the environment [60, 64, 62, 19].

Hähnel *et al.* [26] utilized a piecewise constant tag detection sensor model to first localize the tags and then use the tag map to localize a mobile antenna. Schneegans *et al.* [60] compared histograms of tag detections with previously recorded histograms at different locations. Kleiner *et al.* [37] used a combination of pedestrian odometry and tag detections to perform graph-based RFID SLAM in a large outdoor environment. Kanda *et al.* [36] deployed RFID readers in a science museum and tracked people with attached RFID tags. Vorst *et al.* [80] showed how to learn a tag detection sensor model in a semi-autonomous fashion. Some approaches in the context of WiFi localization model the expected signal strength at different locations by using a discrete grid [64], or Gaussian process regression [62, 19].

Several approaches have addressed exclusively the tag localization problem. Ni *et al.* [48] compared the (emulated) signal strengths of tags at unknown locations with signal strengths received from reference tags at known locations. Alippi *et al.* [1] used several rotating antennas. Ehrenberg *et al.* [15] used an HF RFID system to localize books on a shelf. Liu *et al.* [43] used a tag detection model that is able to estimate the 3D position of the tag. Deyle *et al.* [13] introduced the concept of RSSI images, which are acquired by panning and tilting an RFID antenna attached to a robot while recording the RSSI measurements.

In contrast to the above-mentioned approaches, we model both phenomena – tag detection events as well as signal strength. The increased accuracy of the model allows us to address (a) localization of a mobile sensor relative to given tag locations and (b) mapping of tag locations when these are unknown. Furthermore, we showed how to simultaneously learn the sensor model and estimate the position of the RFID tags in an unsupervised fashion. We presented real-world experiments in an office environment as well as in a supermarket environment and compared our approach with state-of-the-art methods.

3.8 Conclusions

In this chapter, we presented a novel combined sensor model that utilizes both (a) the received signal strength and (b) tag detections of RFID systems for robot localization and mapping of tags. We also presented a technique to learn such a model in an unsupervised way. This greatly simplifies the task of sensor modeling in practice compared to the manual acquisition of calibration data. For comparison, we implemented a sensor model that has been shown to be effective for WiFi localization. We furthermore described how this model can be improved in the context of RFID localization. As our experiments in several real-world settings showed, our approach achieves the computational efficiency of existing sensor-centric models and an accuracy of a state-of-the-art approach that learns a location-dependent model for each tag.

Our proposed technique enables a mobile robot to localize and identify objects. The resulting object map consists of a list of object locations and object IDs. We now move on to the question how robots can take advantage of such object maps and how to leverage knowledge about usual object arrangements to more efficiently carry out their tasks. In particular, we are interested in finding an object in an unknown environment that has to be explored while searching for the object. As a motivating application scenario, we consider the search for a product in unknown supermarket. This will be the topic of the next chapter.

CHAPTER 4

Searching for Objects

We consider the problem of efficiently finding an object with a mobile robot in an initially unknown, structured environment. The overall goal is to allow the robot to improve upon a standard exploration technique by utilizing background knowledge from previously seen, similarly structured environments. We present two conceptually different approaches. Whereas the first method is a reactive search technique that decides where to search next only based on local information about the objects in the robot's vicinity, the second algorithm is a more global and inference-based approach that explicitly reasons about the location of the target object given all observations made so far. While the model underlying the first approach can be learned from data of optimal search paths, we learn the model of the second method from object arrangements of example environments. Our application scenario is the search for a product in a supermarket. We present simulation and real-world experiments in which we compare our strategies to alternative methods and also to the performance of humans.

Consider the situation where you want to find a product in a supermarket where you have never been to before. Certainly, you will not just wander around randomly through the market nor will you systematically visit each so far unvisited aisle in the market until you find the product. Your search will rather be guided by the current observations and the expectations you have about how objects in supermarkets are usually arranged. Over time, you might even have developed some heuristics that proved to be useful for quickly finding a certain product, like "if you want to find yogurt, follow the aisle with the cooling shelves."

The search for a product in an unknown supermarket is a problem that everyone is familiar with and it therefore is an illustrative instance of the kind of search problems we want to tackle with the techniques presented in this chapter. Even for humans [75] this task is not an easy one and we will also compare our search techniques to the performance of human participants that took part in a field study conducted in a real supermarket [35]. However, the supermarket is just an example scenario. Searching for objects or places in offices or domestic environments is conceptually similar. All we assume is that the environment is structured and the object arrangements exhibit some spatial dependencies such that a generalization to an unknown yet similarly structured environment is possible. We regard this as a rather weak assumption that holds for a huge variety of man-made environments. We thus strive for a general way to model, learn, and utilize background knowledge such that a mobile robot is able to find an object more efficiently than it would have been possible without such domain-specific knowledge. For this, we present and evaluate two approaches and provide alternative views on the search problem.

The first approach is a reactive search technique that only depends on local information about the objects in the robot's vicinity when deciding where to search next. This approach emphasizes the sequential nature of the search process, which is a sequential decision making process. Being in a certain state we must choose among a set of available actions. In this setting, background knowledge can be encoded as a state-to-action mapping, a policy, that tells us what to do in a certain situation. In the supermarket scenario, a state includes the currently observed objects in direction of the different aisles and the available actions correspond to the aisle we may choose to visit next. To learn this state-to-action mapping, we draw on ideas from imitation learning [3]. In particular, we want to imitate a simulated robot that exhibits an optimal search behavior by approaching the target object on the shortest path. In each visited state of a demonstrated example path, the robot takes a certain action and discards the other available actions in this state. Thereby, it provides positive and negative examples of state-action pairs to be taken or not, respectively. These examples can be used to learn a classifier for state-action pairs, which yields a classifier-based policy representation [55]. This might either be a multi-class classifier that directly outputs the action to be taken in a given state, or it might be a binary classifier that labels each available action in a state as promising or non-promising (if there is a tie, we may choose randomly among the promising actions). The latter has been empirically shown [55] to yield policies that perform better than the ones that are represented by multi-class classifiers. We use decision trees as binary classifiers which result in compact policy representations that resemble search heuristics like the above-mentioned heuristic for finding yogurt.

The second approach treats the search problem as an inference problem. This is motivated by the observation that we are constantly reasoning about the location of the object while searching for it. This reasoning process will be influenced by the thus far observed objects and structure of the environment as well as our expectations about usual object arrangements in such environments. In the supermarket scenario this means, for example, that if we are searching for beer and in one aisle we observe milk we may conclude that the beer is probably not in the same aisle. In this setting, background knowledge is encoded as expectations about how objects co-occur. However, co-occurrence of objects can only be defined with regard to a spatial context – like objects being "in the same aisle", or one object being "in the neighboring aisle" of the other. Each particular spatial context induces a different local co-occurrence model. In general, there is no single best spatial context and object arrangements in real-world environments are too complex to be faithfully represented by any of these rather basic models alone. Nevertheless, each local model captures useful statistical properties of such object arrangements. Based on these considerations and motivated by the idea of combining an ensemble of base classifiers to form a more robust classifier, we proceed as follows: we use a diverse set of local co-occurrence models, each considering a different spatial relation, and fuse their outcomes as features in a maximum entropy (MaxEnt) model [30, 6] which in our case models the discrete distribution over all possible locations of the target object. The robot then essentially moves to the location which most likely contains the target object. Each time new information becomes available, i.e., newly detected objects or newly discovered parts of the environment, the robot recomputes the distribution.

These two approaches have quite different properties. The first approach uses only local information, as it depends only on the objects in the vicinity of the robot, while the second takes into account all observations made so far. Moreover, the underlying model of the first approach is learned by observing optimal search behavior, while the model of the second is learned from object arrangements of similarly structured example environments.

This chapter is organized as follows. First, we describe our reactive search strategy based on search heuristics. Next, we present the inference-based search strategy as our second approach. We present the results of an experimental evaluation including simulation and real-world experiments. The experiments include a direct comparison of both approaches, as well as a comparison to a baseline strategy and to the search efficiency of human subjects in a real supermarket.

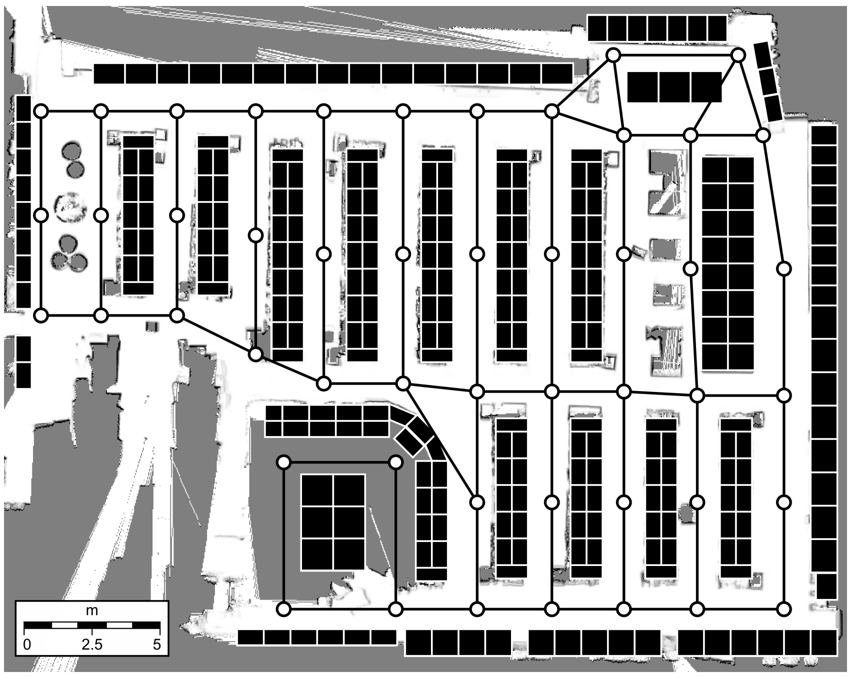


Figure 4.1: Example map of a real supermarket environment. The map contains the shelf locations (black boxes) and the products within the shelves. The underlying structure is a graph. The task of the robot is to efficiently find a certain product.

4.1 Reactive Search Strategy

Both approaches are learned and evaluated on the same data gathered from four real supermarkets, but due to their differences in approaching the search problem, they use different representations of the environment. For example, our reactive search strategy distinguishes aisles into main aisles and side corridors, while for our inference-based search technique this distinction is of no importance. When describing a search strategy, we will therefore also include a section that describes in detail how the search strategies models the environment. In the following, we first describe the reactive search strategy which is based on search heuristics represented as decision trees.

4.1.1 Modeling the Environment

A supermarket $m \in \mathcal{M}$ contains a set of shelves $\mathcal{S}_m \subset \mathcal{S}$ and a graph $\mathcal{G}_m = (V, E)$, as illustrated in Fig. 4.1. Each shelf $s \in \mathcal{S}$ is associated with a location $\boldsymbol{\ell}_s = (x_s, y_s)$ and an orientation θ_s. The relation $\textsc{inMarket} \subset \mathcal{S} \times \mathcal{M}$ associates each shelf with its corre-

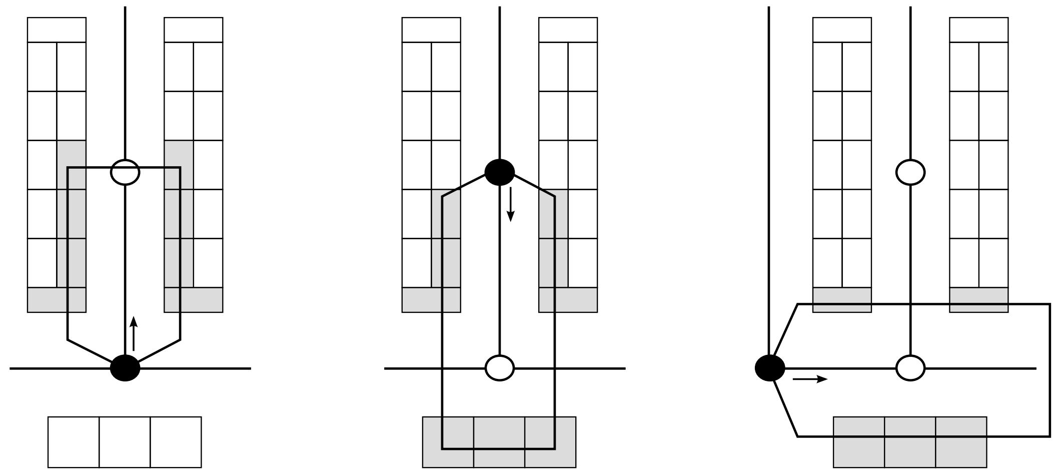


Figure 4.2: Three example situations for illustrating the short range visibility. Whereas gray shelves are visible, white shelves are not visible. The location of the robot is indicated by the black node and its orientation is indicated by the arrow.

sponding market. Furthermore, we define a set of shelf types $\mathcal{T} = \{$NORMAL, COOLING, FREEZER, COUNTER, GROCERY$\}$ and each shelf is associated with exactly one type as defined by the relation TYPE $\subset \mathcal{S} \times \mathcal{T}$. Each shelf contains at least one product and the same product might be placed in several shelves, as defined by the relation INSHELF $\subset \mathcal{P} \times \mathcal{S}$. The relation CATEGOF $\subset \mathcal{P} \times \mathcal{C}$, associates each product with a product category.

For the experiments, we used a set of 196 products at the granularity of small categories like “sugar”, “pizza”, “apples”, “tea”, etc. We furthermore used 20 product categories with a coarser granularity like “breakfast”, “dairy products”, “vegetables & fruits”, etc.

The nodes V of a graph $\mathcal{G}_m = (V, E)$ model the decision points in the supermarket and the directed edges define the reachability between decision points. While the reachability could have been modeled with undirected edges, the visibility of the shelves also depends on the current node (the robot’s current location) and therefore is defined over directed edges. We use two variants of a visibility relation that defines the shelves that are visible when looking into the direction of a certain edge. The first one is a long range variant SHELFVISL $\subset E \times \mathcal{S}$ and the second is a short range variant SHELFVISS $\subseteq$ SHELFVISL. This is motivated by the fact that, although certain information, like the type of a shelf, can be determined reliably over long distances, some information can only be determined when one is in close vicinity to a shelf, like for example the products contained within a shelf. Three example situations illustrating the short range visibility are depicted in Fig. 4.2. On the basis of the two visibility relations we define several other visibility relations, like the visible products

$$\textsc{prodVis} = \{(e,p) \mid \textsc{shelfVisS}(e,s), \textsc{inShelf}(p,s)\}, \tag{4.1}$$

and the visible product categories

$$\textsc{categVis} = \{(e,c) \mid \textsc{prodVis}(e,p), \textsc{categOf}(p,c)\}. \tag{4.2}$$

The visibility of shelf types is modeled in such a way that we can distinguish whether the shelf type is seen in the direct vicinity, or being observed at a further distance:

$$\textsc{typeVisS} = \{(e,t) \mid \textsc{shelfVisS}(e,s), \textsc{type}(s,t)\} \tag{4.3}$$

$$\textsc{typeVisL} = \{(e,t) \mid \textsc{shelfVisL}(e,s), \textsc{type}(s,t)\}. \tag{4.4}$$

The reactive search strategy utilizes the information associated with each edge to decide which edge to follow. In the next section we describe how we learn such a strategy from data by observing optimal search paths.

4.1.2 Learning Search Heuristics

We are interested in learning a reactive search strategy that depends only on local information in order to find a certain target product. We therefore classify the outgoing edges of the current node by a decision tree into promising and non-promising directions based on the information associated with each edge. For learning such a decision tree, we first need to define appropriate edge attributes and then gather training data by generating optimal search paths in a training set of supermarkets. To evaluate the strategy, we apply it to a previously unseen market.

4.1.2.1 Defining Edge Attributes

One obvious piece of information, by which the search should be guided, is which products and product categories are visible at a certain edge. If we are searching for coffee and an aisle contains tea, or in general breakfast products, then this edge is certainly a promising candidate. But the decision should also be influenced by additional factors. If we know that an edge has been visited already, we can reject it in order to avoid loops. Also the type of an edge might be of interest, such as if an edge belongs to an aisle that follows a wall

(wall aisle), because some products, like milk, are only located in such aisles. Likewise, we define main aisles as aisles that follow a main direction in a market and from which many narrow side corridors branch off. Next, it is informative if the robot is approaching certain landmarks in the supermarket, like the entrance, the exit, or the back of the market. Vegetables, for example, are always located near the entrance in our markets. Thus, each optimal search path for finding apples would mostly contain edges that are approaching the entrance. Likewise, frozen food is usually in the back of the market and wine and non-food are near the exit of the market.

We also use statistics about the expected relative product position between the entrance and the exit based on the layout of all training markets. The relative position of a shelf s with respect to the location $\boldsymbol{\ell}_{en}$ of the entrance node and the location $\boldsymbol{\ell}_{ex}$ of the exit node of the corresponding market is defined as

$$\mathrm{relPos}(s) = \frac{\|\boldsymbol{\ell}_s - \boldsymbol{\ell}_{en}\|}{\|\boldsymbol{\ell}_s - \boldsymbol{\ell}_{en}\| + \|\boldsymbol{\ell}_s - \boldsymbol{\ell}_{ex}\|}. \tag{4.5}$$

The expected relative position of a *product* is then defined as the average of these values for all shelves that contain this product in the training markets M_t

$$S_p = \{s \mid \textsc{inShelf}(p, s), \textsc{inMarket}(s, m), m \in M_t\} \tag{4.6}$$

$$\mathrm{expRelPos}(p) = \frac{1}{|S_p|} \sum_{s \in S_p} \mathrm{relPos}(s). \tag{4.7}$$

We define a binary edge attribute (No. 222 in Tab. 4.1) that indicates if the robot would be approaching the expected relative position of the target product by following that edge.

Furthermore, we calculate the average Euclidean distance $\mathrm{prodDist}(p_i, p_j)$ for each pair (p_i, p_j) of products based on their locations in the training markets. If we denote by $P_{i,j}$ the set of visible products that are associated with an outgoing edge $e_{i,j}$ from the current node v_i to a possible successor node v_j, then the average product distance of this edge to the target product p_t is defined as

$$\mathrm{avgProdDist}(e_{i,j}, p_t) = \frac{1}{|P_{i,j}|} \sum_{p \in P_{i,j}} \mathrm{prodDist}(p_t, p). \tag{4.8}$$

We define an indicator attribute (attribute No. 223 in Tab. 4.1) that is set to true if an edge has the lowest average product distance of all outgoing edges of the current node, and thus can be considered to be the most promising edge with respect to the expected product

Table 4.1: The attributes that are used to characterize an edge. All attributes are binary. In the experimental evaluation we test different combinations of subsets (a–d) of these attributes.

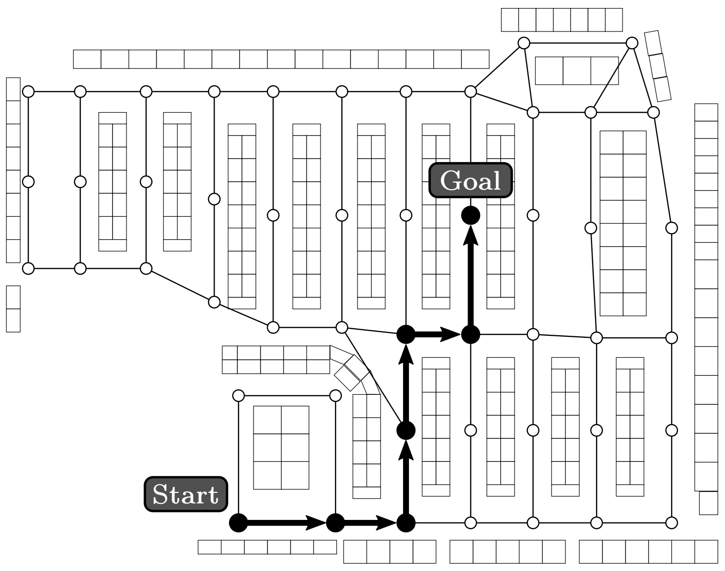
Subset	Att. No.	Description
a	1	Edge already visited
a	2–197	Product $p_i \in \mathcal{P}$ visible
a	198–217	Product of category $c_i \in \mathcal{C}$ visible
a	218	Shelf of type NORMAL visible (short range)
a	219	Shelf of type COOLING visible (short range)
a	220	Shelf of type FREEZER visible (short range)
a	221	Shelf of type COUNTER visible (short range)
b	222	Leads to expected relative position
b	223	Has smallest avg. Euclidean distance to product
b	224	Has smallest avg. path distance to product
c	225	Current node belongs to a main aisle
c	226	Next node belongs to a main aisle
c	227	Next node belongs to a wall aisle
c	228–230	Leads to the entrance, exit, or back of the market
d	231	Shelf of type COOLING visible (long range)
d	232	Shelf of type FREEZER visible (long range)
d	233	Shelf of type GROCERY visible (long range)

distances. Likewise, we define an attribute that uses the path distance on the graph between products instead of the Euclidean distance (attribute No. 224). As it is not easy to decide beforehand whether the path distance or the Euclidean distance is a more reliable indicator for product distances we use both attributes and let the learning algorithm decide which one to use during the induction of the tree. A complete list of all attributes can be seen in Tab. 4.1.

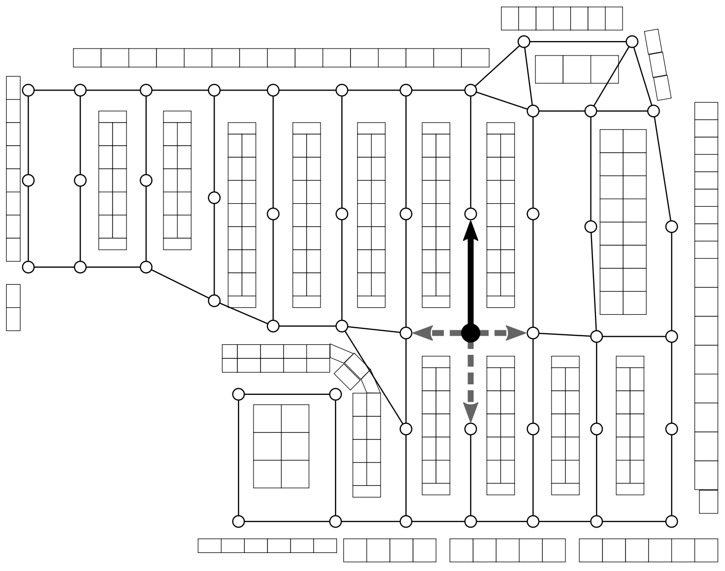
4.1.2.2 Generating Training Data

We use a fixed set of 15 target products. These are the same products that human participants had to find in a field study conducted in the very same supermarket in which we will evaluate our strategy. We learn a separate decision tree for each of these 15 target products.

We determine for each node in a training supermarket the shortest path to a given target product. Each node of an optimal path corresponds to a local decision for taking a certain outgoing edge (the one that leads to the next node of the optimal path) and for rejecting all other outgoing edges of that node. In this way, each optimal search path contributes a set of positive and negative examples of edges to be taken or not, respectively. Fig. 4.3 illustrates the basic idea. The positive and negative examples of all paths for all starting positions in



(a) Shortest path between starting location and goal location.



(b) Local decision at one of the nodes along the path.

Figure 4.3: (a) For generating training data, we compute the shortest path from every possible starting location in the market for a given goal location of a target product. One such path is depicted above. (b) Each node of an optimal path corresponds to a local decision for taking a certain outgoing edge (black solid arrow) and for rejecting all other outgoing edges of that node (gray dashed arrows). In this way, each optimal search path contributes a set of positive and negative examples of edges to be taken or not, respectively.

all training supermarkets then constitute the training data for learning the decision tree for a given target product.

As there might exist more than one optimal path from a starting location to the target location, we search for more than just a single shortest path to generate training data. Additionally, as the decision points are placed manually, there might be small differences between nearly optimal paths. For this reason, we also generate training data from paths, which are not longer than a given small threshold when compared to the actual shortest path.

4.1.2.3 Decision Tree Learning and Pruning

We use the well known ID3 algorithm [52] to learn a decision tree. As decision trees can be prone to overfitting, we will therefore also investigate the influence of two pruning techniques in the experimental evaluation. The two techniques are a restriction on the maximum depth of the tree (max-depth-pruning, MDP) and reduced error pruning (REP) [8]. In MDP, every subtree that has its root node at a given depth of the original tree will be collapsed into a leaf node. For REP, we need to divide the training data set into an induction set, which is used during induction of the decision tree, and a pruning set, which is used to evaluate which part of the tree should be pruned. REP then replaces any subtree with a leaf node if this does not lead to a higher classification error on the pruning data set.

The learned decision tree is then used to guide the search for the target product by classifying each outgoing edge of the robot's current location into promising and non-promising directions. It may happen that more than one edge will be classified as promising. In this case, we choose randomly among the promising candidates. If there are no promising edges, we randomly choose among the unvisited edges. If all outgoing edges have been visited already, the robot moves on the shortest path to the nearest known node, which has at least one unvisited outgoing edge.

4.1.3 Variants of the Decision Tree Strategy

In total, we evaluate five variants of the decision tree strategy. The first four variants differ by the set of attributes they are allowed to use. We start from a simple variant, which uses only subset "a" of the attributes (see Tab. 4.1), while the three subsequent variants can use increasingly more attributes (including subsets "b", "c", and "d"). The resulting decision trees are not pruned in any way and therefore might be prone to overfitting. We therefore also investigate the influence of the two above-mentioned pruning techniques. We tried several alternatives by restricting the maximum depth of the trees to different levels (MDP) or by applying reduced error pruning (REP), or a combination of both to any of the four attribute subset variants. We found the best variant to be a combination of both pruning techniques applied to a tree that uses the full set of attributes. We first applied MDP using a maximum depth of four and then additionally applied REP. To do so, the training data set was split into an induction set (75% of the data) and a pruning set (25% of the data). Two examples of learned and pruned decision trees can be seen in Fig. 4.4 on the next page.

4.2 Inference-based Search Strategy

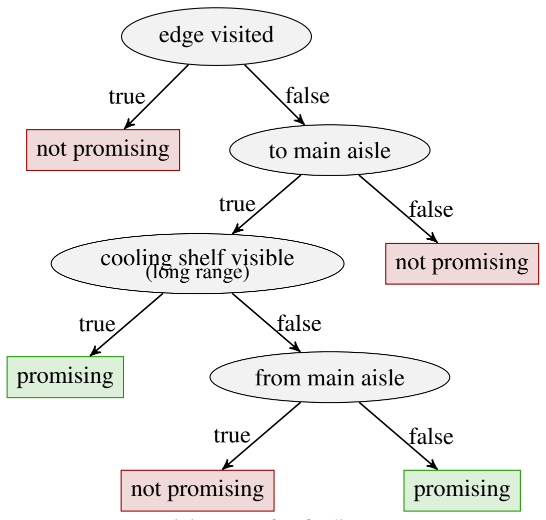
In this section, we describe our second approach, the inference-based search strategy. We focus on the problem of how background knowledge about usual arrangements of objects can be utilized to more efficiently find an object. The basic idea of this approach is to split the action selection problem during the search process into two parts. First, we compute a belief over possible locations of the target object based on the information about the objects seen so far and the structure of the environment. Second, we utilize the belief to select the next action, e.g., the next position the robot moves to.

The representation for encoding the background knowledge about object arrangements is motivated by the fact that structured indoor environments exhibit meaningful spatial relations between locations that influence the distribution of the objects. For example, one might link two locations by relations like “the same room”, or “the same floor”. Furthermore, we think that in order to be able to generalize from previously seen environments to new environments it is advantageous to represent objects by a list of attributes, instead of thinking of them as atomic entities. For example, even if you have never seen an avocado in a supermarket, it will be useful to know that it is a fruit and it is therefore probably located somewhere where you will see other objects of the category “fruit”.

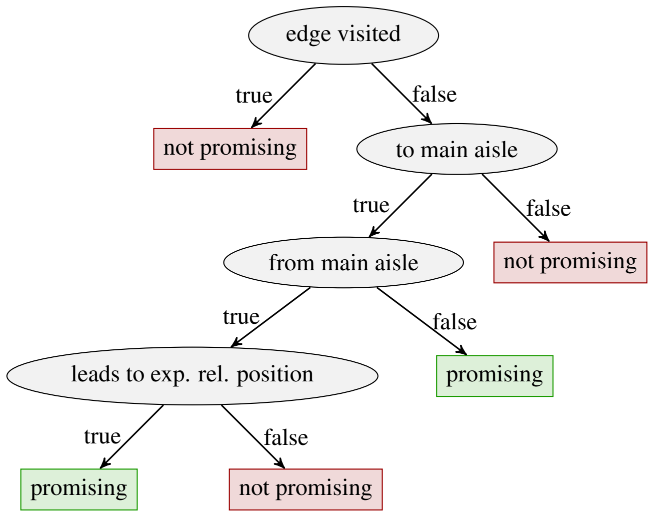
The definition of the object attributes and the types of spatial relations are the only domain-specific parts of this second approach. It is therefore also applicable to other application scenarios, like finding an object in an office environment or a domestic environment as long as it is provided with the corresponding relevant attributes and spatial relations.

4.2.1 Modeling the Evironment

We collected real-world data from four supermarkets, including the market layout and the products in each shelf. We defined a set of 181 products at the granularity of small cate-



(a) Decision tree for finding yogurt.



(b) Decision tree for finding UHT milk.

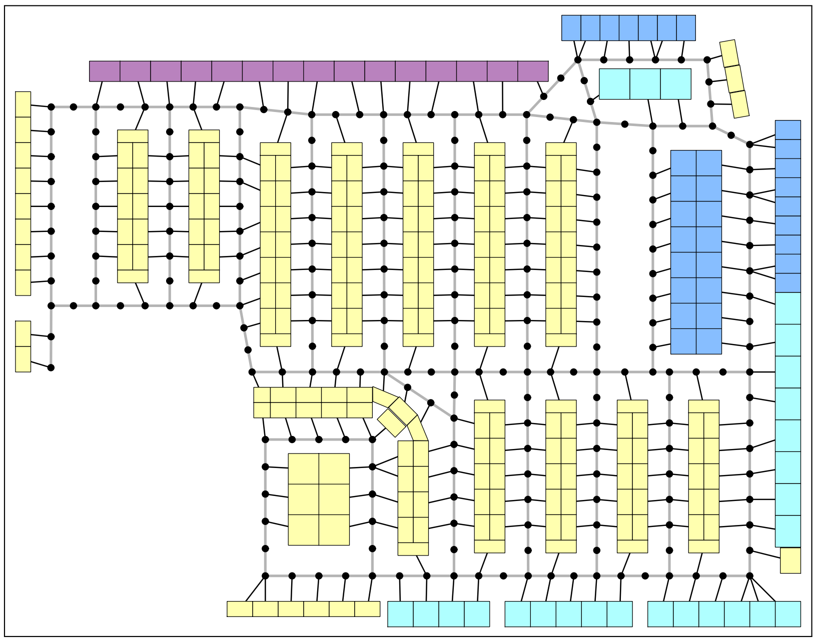
Figure 4.4: Two examples of pruned decision trees that have been learned from optimal search paths in the training supermarkets. The trees use the attribute variant (a–d, pruned) mentioned in Tab. 4.2.

gories like pizza, apple, shampoo, etc. Additionally, each product is associated with one of 20 product categories with a coarser granularity like breakfast, dairy products, vegetables & fruits, etc. Further attributes of products are the binary "edible" attribute, as well as the attribute "shelf type", that denotes in which shelf type the product is usually found in. This can be normal, cooling, freezer, or counter. We consider the basic structural elements of a supermarket to be approximately one meter wide shelves, which are depicted as boxes in Fig. 4.5 and Fig. 4.6 on the next pages. Moreover, we will define a "shelf wall unit" to be made up of adjacent shelves standing side by side, such as the red region marked in Fig. 4.8 on page 89. Each shelf contains at least one product and a product might be placed in several shelves.

Additionally, the representation of a market contains a graph that constrains the motion of the robot. Each shelf is associated with an access node which is defined to be its nearest graph node. The search process ends, if the robot is located at the access node of a shelf that contains the target product. The robot does not know the environment beforehand but rather has to explore it during the search. The structure of the environment – shelves and graph nodes – can be observed from any distance within the market, as long as they are in the line of sight. In contrast, the products of a shelf can only be detected within a distance of two meters. These visibility constraints are motivated by taking into account the sensor limitations of a real robot. The graph could be extracted from a grid map by constructing a Voronoi graph and the shelf locations could be identified by assuming that each wall or obstacle in the map is a shelf. For the actual detection of the objects, we assume that the robot is equipped with an RFID sensor and the products are equipped with RFID tags. In a supermarket environment the robot would then be able to reliably locate a product when it is about two or three meters away. However, the model described here is not restricted to these sensor modalities, we only require the robot to be able to sense the structure of the environment and to detect and localize objects. The next section will introduce our model for inferring the location of the target object that the robot is searching for.

4.2.2 A Model for Inferring Object Locations

We have a set $O = \{o_n\}_{n=1}^{N_o}$ of detectable objects and each object is described by a set $\mathcal{A} = \{A_i\}_{i=1}^{N_i}$ of attributes, each with a finite domain $D(A_i)$ of possible values $a_i \in D(A_i)$. We wish to infer the location of a query object $o_q \in O$, which we assume to be at one of several possible locations $X = \{x_l\}_{l=1}^{N_l}$. We furthermore have a set of spatial relations $R = \{r_j\}_{r=1}^{N_r}$, that relate locations of detected objects to locations $x \in X$, like "same


(a) Supermarket environment.

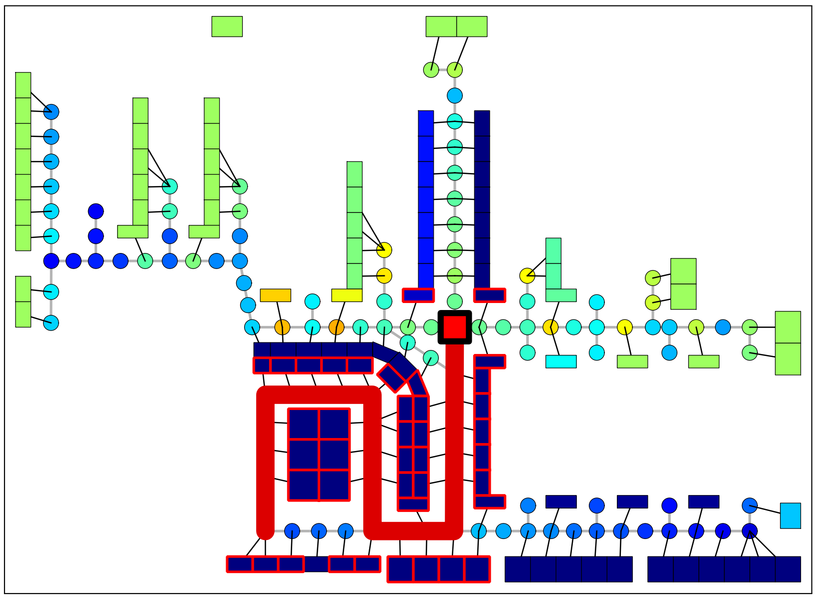

(b) Searching for a product.

Figure 4.5: (a) We model supermarket environments including shelves, products, and a graph. (b) The simulated robot (red square), which does not know the environment, has to search for a product. The search is adapted according to the robot's belief over the possible product locations, which takes background knowledge about usual object arrangements into account.

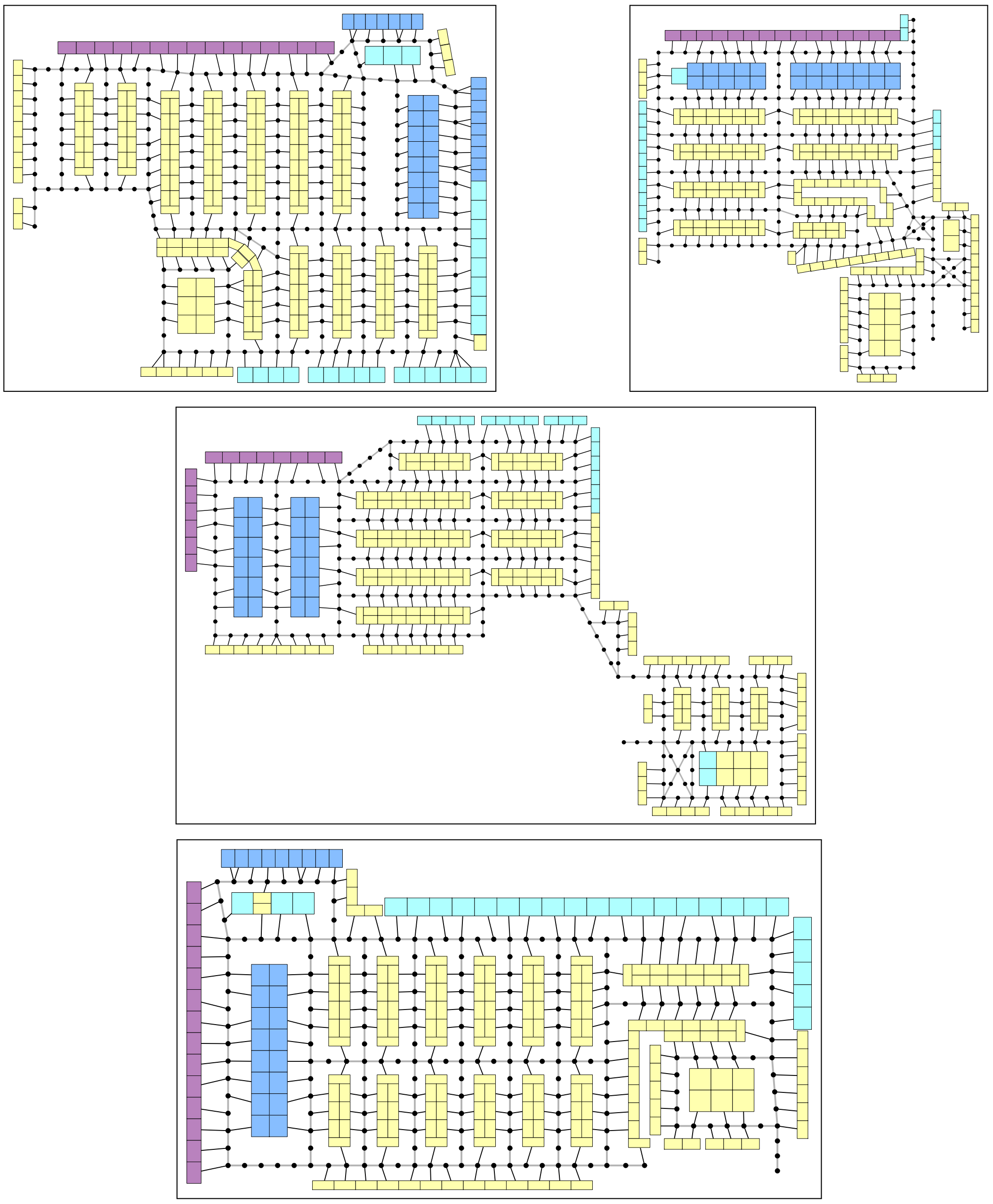


Figure 4.6: We collected real-world data from four supermarkets, including the market layout, the shelf types, and the products in each shelf. The colors indicate the different shelf types, like normal shelf (yellow), cooling shelf (light blue), freezer (dark blue), or counter (purple).

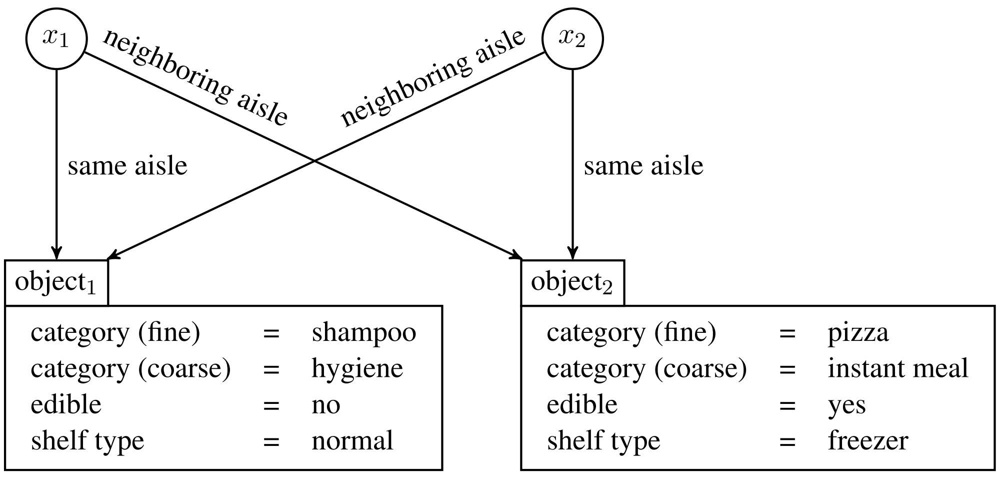


Figure 4.7: Illustration of the basic idea for inferring object locations based on the detection of object attributes in different spatial contexts. Possible locations of the query object we are searching for are linked to already seen objects by different spatial relations.

room", or "same aisle". We also allow the definition of overlapping relations, like "same room" and "same house". The location of a detected object might be linked to several locations x_l by different spatial relations r_j. See Fig. 5.3 for an illustration of the basic idea. An observation z in our model corresponds to a tuple $z = (x_l, r_j, A_i, a_i)$ that states that an object with the attribute $A_i = a_i$ has been observed at a location that is related to x_l by the spatial relation r_j. For example, in the supermarket x_l may denote a certain shelf and the robot could make an observation $z = (x_l, r_j =$ "same aisle"$, A_i =$ "category (fine)"$, a_i =$ "pizza"$)$, which corresponds to the situation that the robot has observed an object of category "pizza" in the same aisle as shelf x_l. But the same detection event would also generate the observation $z = (x_l, r_j =$ "same aisle"$, A_i =$ "edible"$, a_i =$ "yes"$)$, because by discovering the pizza it thereby has also discovered an edible product. Thus, a single newly detected object introduces several basic observations z_i, because an object is described by several attributes and is related to several other locations in the supermarket by several different spatial relations. Next, we denote by $\mathbf{z}$ all observations made so far and $\mathbf{z}_{[x_l, r_j, A_i]}$ denotes the subset of observation that are constraint to have values x_l, r_j, and A_i.

Throughout the search process, we must update the belief $p(x \mid \mathbf{z})$ over possible locations $x \in X$ of the query object o_q that we are searching for, given the observations $\mathbf{z}$ made so far. To model $p(x \mid \mathbf{z})$, we rely on background knowledge about co-occurrences of objects and object attributes in different spatial contexts. This knowledge is expressed by conditional probability distributions $p(A_i = a_i \mid o_q, r_j)$ that specify the probability of the following event: given that object o_q exists at some location, then there will exist another object with attribute $A_i = a_i$ at any location that is related to the location of o_q by the spatial relation r_j. For example, we might ask that under the assumption that the “coffee cup” that we are searching for is in room x, what would be the probability that we observe a “kitchen object” in “the same room” as room x. Additionally, we need to model $p(A_i = a_i \mid \neg o_q, r_j)$ that we will see the attribute in a related location, given the object is *not* present.

We follow a two step process to compute the desired final distribution $p(x \mid \mathbf{z})$ based on the above-mentioned conditional distributions. The idea is to use an ensemble of local models, each considering only a certain aspect of the observations, and then to fuse the local models in a combined model that computes the final distribution. This is motivated by the assumption that the distribution of objects in real-world environments is too complex to be faithfully captured by just a single model and it therefore would be beneficial to combine a diverse set of more simple models. These local models compute the binary probability $p_{A_i,r_j}(x \mid \mathbf{z})$ that the object exists at location x versus that it does not exist at this location $p_{A_i,r_j}(\neg x \mid \mathbf{z})$. Each local model considers only a certain attribute A_i of those observations $\mathbf{z}_{[x,r_j,A_i]}$ that are related to x by the relation r_j.

For example, in the supermarket a local model $p_{A_i,r_j}(x \mid \mathbf{z})$ computes the probability for the presence or absence of the sought product in a certain shelf x. For computing this probability, this local model considers all observations generated by objects that are related to x by a certain spatial relation – for example, that are in the same aisle as x and hence the model would consider all observations that have $r_j =$ "same aisle". The objects that generated these observations are described by several attributes, but the local model only takes into the information conveyed by a certain attribute A_i, for example, $A_i =$ "category (coarse)". In essence, such a model would compute the probability for the presence or absence of the sought product in shelf x based on the fact that there are a certain number of “instant meal” products, “hygiene” products, etc., in the same aisle as shelf x.

Now let $\mathbf{a}(\mathbf{z})$ denote the set of attribute values that occur in the observations $\mathbf{z}$. Then we model the local models as binary naive Bayes classifiers as follows:

$$p_{A_i,r_j}(x \mid \mathbf{z}) = \frac{p(x) \prod_{z \in \mathbf{z}_{[x,r_j,A_i]}} p(z \mid x)}{\sum_{x' \in \{x, \neg x\}} p(x') \prod_{z \in \mathbf{z}_{[x,r_j,A_i]}} p(z \mid x')} \tag{4.9}$$

$$= \frac{\prod_{a \in \mathbf{a}(\mathbf{z}_{[x,r_j,A_i]})} p(A_i = a \mid o_q, r_j)}{\sum_{o' \in \{o_q, \neg o_q\}} \prod_{a \in \mathbf{a}(\mathbf{z}_{[x,r_j,A_i]})} p(A_i = a \mid o', r_j)}. \tag{4.10}$$

In (4.10) we dropped the prior $p(x)$, which we assume to be uniform. In total we will have 20 local models as we will be using four attributes and five relations, which we introduce in the next section. The output of all local classifiers p_{A_i,r_j} will be used as features f_{A_i,r_j} in the maxent model:

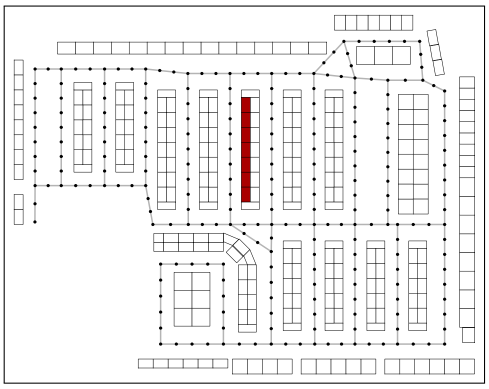
$$p(x \mid \mathbf{z}) = \frac{\exp\left(\sum_{A_i,r_j} \lambda_{A_i,r_j} f_{A_i,r_j}(x,\mathbf{z})\right)}{\sum_{x'} \exp\left(\sum_{A_i,r_j} \lambda_{A_i,r_j} f_{A_i,r_j}(x',\mathbf{z})\right)}. \tag{4.11}$$

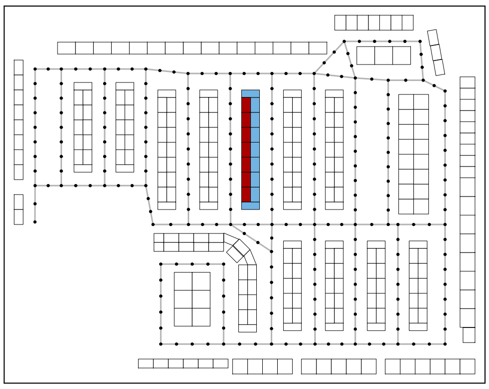
Thus, the maxent model is used as a way to combine an ensemble of base classifiers. In the experimental section we will also evaluate two other methods for combining the local models. The first method is a weighted average $p(x \mid \mathbf{z}) \propto \sum_i \lambda_i P_i(x \mid \mathbf{z})$, which is also known as the linear opinion pool. The second model is the logarithmic opinion pool $p(x \mid \mathbf{z}) \propto \prod_i P_i(x \mid \mathbf{z})^{\lambda_i}$ which applies exponential weights and corresponds to the geometric mean if the weights are uniform and normalized [27, 65].

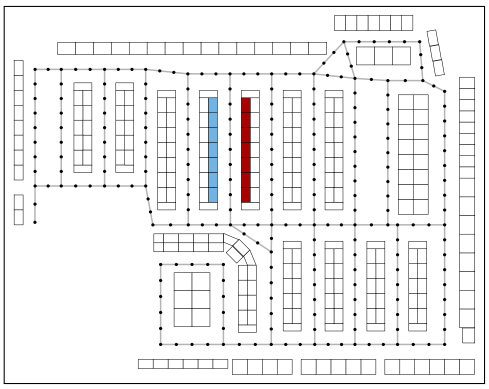
There are several possibilities of relating the probabilities $p_{A_i,r_j}(x \mid \mathbf{z})$ of the local models with the features f_{A_i,r_j} of the maxent model. One option is to use the probabilities directly as the features. However, continuous features are usually discretized when used in a maximum entropy approach. Thus, we will also consider to discretize the probabilities in four equally sized bins. Each original feature is then represented by four binary features, of which only exactly one feature can be non-zero at a time, depending on the probability of the local model. A third option that we consider is to use the log-probability of the local models – in this case the fusion of the local models resembles a logarithmic opinion pool [27, 65], because $\exp(\sum_i \lambda_i \log(P_i)) = \prod_i P_i^{\lambda_i}$.

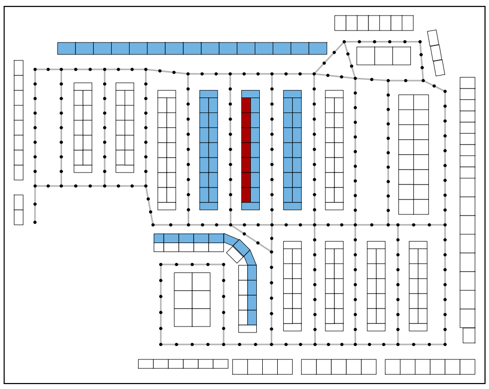
4.2.2.1 Application to the Supermarket Scenario

We wish to infer in which shelf wall unit the target product is located in, given the products seen so far. To apply our model to this scenario, we need to define the object attributes and the spatial relations that we consider to be meaningful in a supermarket. Fig. 4.8 illustrates the five relations between shelf wall units that we will be using. This includes the relation "same unit", as well as different types of neighborhood relations, like a unit that shares an access node on the graph with the reference unit, or that is adjacent to the reference unit. Further, we consider relations based on the Euclidean distance and the path distance between shelf wall units. Each relation is reflexive and thus also includes the reference unit.


(a) Same unit.


(b) Adjacent unit.


(c) Shared access node unit.


(d) Short Euclidean distance unit.

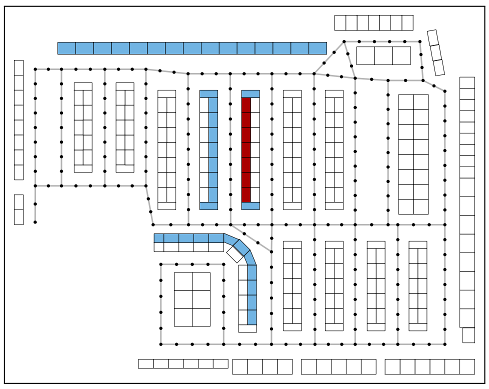

(e) Short path distance unit.

Figure 4.8: We use five relations between shelf wall units. The reference unit is marked in red (dark gray) and the related units in blue (light gray). The relations are: (a) The same shelf wall unit. (b) Units adjacent to the reference unit. (c) Units sharing an access node with the reference unit. (d) Units within an Euclidean distance of four meters, or (e) within a path distance of four meters. All relations are reflexive and thus also include the reference unit.

The attributes we are using are the ones that already have been depicted in Fig. 5.3 on page 112. These are: fine category, coarse category, shelf type, and edible. Some examples of the fine categories (pizza, shampoo, etc.) and coarse categories (instant meal, hygiene, etc.) that we are using were given in Section 4.2.1 on page 81.

To learn the feature weights we set up the training data in the following way: first we estimate the conditional probabilities $p(A_i = a_i \mid o_q, r_j)$ based on the data of three supermarkets by simply counting the basic events. The local models are then used in the maximum entropy model as features to predict the locations of objects in the remaining fourth supermarket. This is done for all four supermarket combinations and for all 141 products, that are available in all markets. Thus, in total we have 4×141 training examples. We train a single set of parameters. The learned weights therefore reflect the general importance of each local model when being used to predict a new situation – independent of a specific product or market. During the search process, we will only observe a small fraction of all products in the market. The model is therefore trained on markets in which only some of the shelves contain products. It takes less than two minutes to train the model on a standard PC and each inference during the search takes less than 10 ms.

4.2.3 Selecting a Target Location

Once a belief over the possible location of the target object is computed, the robot needs to decide which action to take next. However, we are facing two problems, when planning a path based on the current belief. First, ideally we should take future observations into account, which will change the belief and the subsequent steps. Second, even if we ignore for now that new evidence will change this belief, the problem of planning an optimal search path that minimizes the expected search path length with respect to a given distribution is still NP-hard [77, 5]. We will thus settle for a heuristic approach.

If the robot greedily plans the shortest path to the node with the highest detection probability, it will exhibit undesirable oscillating behavior when there is more than one mode in the belief. We therefore use a strategy that computes a utility $U(v)$ for each node v that trades off the (relative) detection probability $p(v)$ at this node with the (relative) path distance $d(v)$ needed to reach it:

$$U(v) = \alpha \frac{p(v)}{\max_{v'} p(v')} - (1-\alpha) \frac{d(v)}{\max_{v'} d(v')}. \tag{4.12}$$

Selecting a target location by trading off the benefits and costs in a weighted sum has been

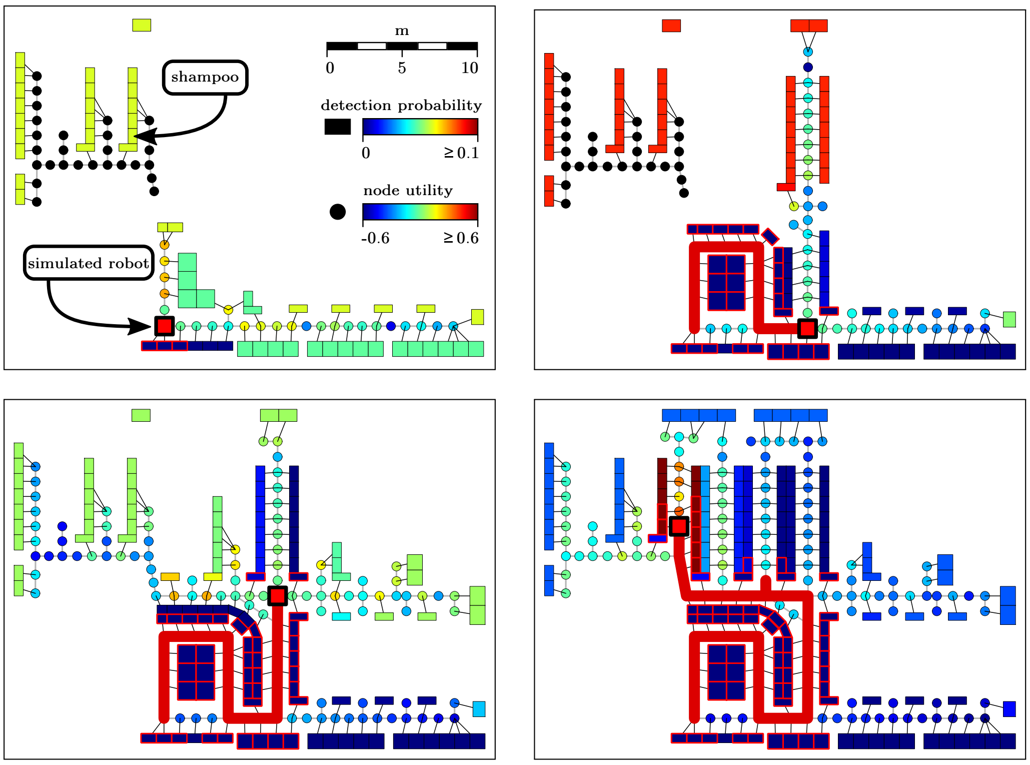


Figure 4.9: Example run using the maxent search technique with log-probability features. The position of the robot is marked by the big red square and the path taken is marked by the red line.

previously used in autonomous exploration [44, 67]. By adjusting the parameter $\alpha \in [0, 1]$ we can alter the search behavior. We tried several parameters in 0.1 increments and found $\alpha = 0.4$ to result in the shortest paths. The robot moves on the shortest path to the unvisited node with the highest utility until new observations are made and the belief and the node utilities have to be re-evaluated. In Fig. 4.9 we illustrate several stages of an example run in one of the supermarkets. This concludes the description of our second search strategy proposed in this chapter. The next section presents the results of several experiments to quantitatively evaluate both strategies.

4.3 Experimental Evaluation

The first experiment is aimed at comparing the performance of the different search strategies in comparison to the performance of humans searching in a real supermarket environment. The second experiment is aimed at a more thorough evaluation of the search strategies, though we do not have data from human participants for this setting. A third experiment presents the results obtained by real-world experiments in which a robot autonomously searched for a product using the decision tree strategy. Finally, we will present further evaluations of the more efficient inference-based search strategy.

The supermarket data, including the layouts and the product placement, was collected in real supermarkets. Three of the supermarkets were used as a training set for learning the underlying models of the search strategies, such as the decision trees and the local models, and the fourth supermarket was used for evaluation of the strategies.

As a baseline approach, we use an exploration strategy that selects randomly among the unvisited edges at the current node. If all outgoing edges of a node have been visited already, an edge will be chosen that leads to the nearest node with at least one unvisited edge. This strategy rapidly explores unvisited areas and avoids searching the known parts of the environment. This is akin to the frontier-based exploration strategy [82] known in mobile robot exploration. If a search technique does not perform better than the exploration technique, it obviously is not able to utilize domain-specific information, which is the ambition of our strategy. As a baseline approach, the exploration strategy allows us to relate the performance of the search strategies to an expected upper bound on the search path length, defined by the average search path length of the exploration strategy. A strict lower bound is defined by the shortest possible path.

4.3.1 Evaluation in Comparison to Humans

A field study involving 38 human participants was conducted in a real supermarket [35]. The participants had to find 15 products in a given order and we used the same 15 products as target products in our simulated search. As the supermarket in which the study took place was the same market that we used as a model for our evaluation market, we can compare the path distances of the human participants to the path distances traveled by the robot in the simulated environment. In order to assure that we have a metrically comparable model of the real market, we first built an occupancy grid map of the supermarket using a laser-based FastSLAM implementation [25] and then placed the shelves according to the grid-map, as can be seen in Fig. 4.1 on page 74. The product placement in our virtual markets also resembles the product locations in the real markets. The participants were tracked using a RFID-based localization technique [33] and the resulting trajectories were then mapped upon the graph for a fair comparison with the path distances of the simulated robot.

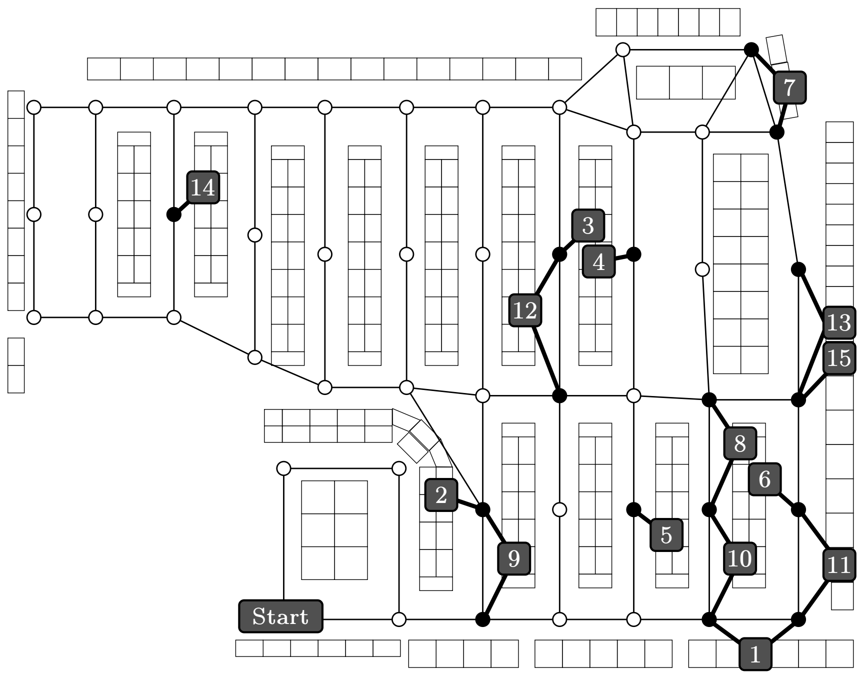


Figure 4.10: In the first experiment the simulated robot and the human participants had to find 15 products in a given order. They start at the entrance of the market in the lower left corner. The product locations are connected to their respective target nodes on the graph. If the robot enters a target node of the current product, the product is found and the robot will search for the next one.

The human participants had to find the 15 products in a given order shown in Fig. 4.10, and so the location of a found target product was the starting location for the search for the next target product. Therefore, each target product was associated with a certain starting location and we evaluated the *simulated* search strategies for the same 15 pairs of starting location and target product.

As a performance measure we consider the length of a complete search path, that is the path length of a search for all 15 products. We simulated a thousand search trials for the exploration strategy and the decision tree strategies. The MaxEnt search strategy is deterministic. We only have a sample size of 26 complete search trials of the human participants, because some search sub-trials (for a single product) have been canceled if the search took too long or the participants gave up. This introduces a slight bias to the comparison for the benefit of the human participants, because the *simulated* search trials were not canceled if they took “too long”. Nevertheless, we think that the available data of the human search paths still constitutes a usable basis for a comparison.

Table 4.2: Mean and standard deviation (SD) of the overall search path lengths for different search strategies. For further comparison we list the length of the optimal path and the path length ratio defined as the average path length of a strategy divided by the length of the optimal path.

Strategy	Search Path Length		Ratio	Samples
	Mean (km)	SD (km)		
Exploration	1.959	0.297	7.9	1000
Dec. Tree (a–d, pruned)	1.176	0.211	4.8	1000
Dec. Tree (a)	1.609	0.263	6.5	1000
Dec. Tree (a–b)	1.425	0.193	5.8	1000
Dec. Tree (a–c)	1.620	0.257	6.6	1000
Dec. Tree (a–d)	1.717	0.238	7.0	1000
MaxEnt (single run)	0.342	–	1.4	–
MaxEnt (restarts)	0.911	–	3.7	–
Human Participants	0.565	0.110	2.3	26
Optimal Path	0.247	–	1.0	–

In Tab. 4.2 we present the mean and standard deviation of the search path lengths. We performed a one-tailed paired t-test[1] for the sampled strategies and found all improvements indicated by the means to be significant at the 0.01 level, except for the difference between the decision trees with attribute combinations (a) and (a–c).

The exploration strategy yielded search paths that are on average 7.9 times longer than the optimal path. This can be improved to a ratio of 4.8 when the search was guided by our proposed strategy based on the pruned decision trees. If we used unpruned decision trees then the best ratio we achieved was 5.8. This seems to suggest that the unpruned decision trees overfit the data of the three training supermarkets. The MaxEnt strategy achieved the best result with a ratio of 1.4. However, by using the MaxEnt strategy the robot built up a global map during the search that also included the locations of all products seen so far. Thus, if a target product has been seen previously while searching for another product, the product could later be approached directly on the shortest path. As the other strategies lack the ability to memorize the global locations of previously seen products, we also considered a modified version of the MaxEnt strategy (the "restart" version in Tab. 4.2) in which the map was cleared when a product has been found. Thus, when searching for the next product on the list, the robot could not utilize information about the market stemming

[1] If the sample sizes differed, we used the sample size of the smaller sample. We also applied Welch's t-test, which is applicable for unequal sample sizes and unequal variances, and got the same results regarding the statistical significance at the 0.01 level.

from the search sub-trial for the previous product. This resulted in a ratio of 3.7, which is still better than the decision tree strategy, though it now performs worse than the human participants who achieved a ratio of 2.3. However, one might argue that humans certainly do build up some kind of global map of the market during the search and thus are able to utilize information from previous search sub-trials when searching for the next product. Clearly, this information is less accurate and more sketchy than the map utilized in the "single run" version of the MaxEnt strategy that memorizes the exact location of all seen products.

Though the proposed strategies did not achieve the same performance as humans, the results clearly indicate that the utilization of background knowledge by our proposed strategies leads to significantly shorter search paths when compared to an uninformed search strategy. The exploration strategy performed significantly worse than our approaches, because it is not able to take domain-specific background knowledge into account, which is the advantage of our proposed techniques.

4.3.2 Evaluation with Varying Starting Locations

The setting presented in the previous section was restricted to a single starting location for each target product. This was motivated by the desired comparison to the performance of the human participants, for which we had to replicate the conditions of the field study.

For a more thorough evaluation of the search strategies, we started the search for the target products from several different locations. We randomly chose 20 starting locations in the market. These are depicted in Fig. 4.11 on the next page. Each of the 15 products from the previous experiment had to be searched for from each starting location, yielding 300 independent search trials per strategy. For the MaxEnt strategy, the map was cleared between the individual search trials. As in the previous experiment, we considered the total search path length of all individual search trials as a performance measure for the search strategies. We repeated each experiment a thousand times and list the average total search path length and its standard deviation in Tab. 4.3 on the next page.

Beside the exploration strategy we evaluated the variant of our decision tree strategy that performed the best in the previous experiment. Though the improvement over the exploration strategy was now less pronounced than in the previous setting, it still yields significantly[2] shorter search paths, as can be seen in Tab. 4.3. The MaxEnt strategy yields

[2]We performed one-tailed paired t-tests with $p < 0.01$ for the sampled strategies. The result of the MaxEnt strategy also differs significantly ($p < 0.01$) from the results of the sampled strategies.

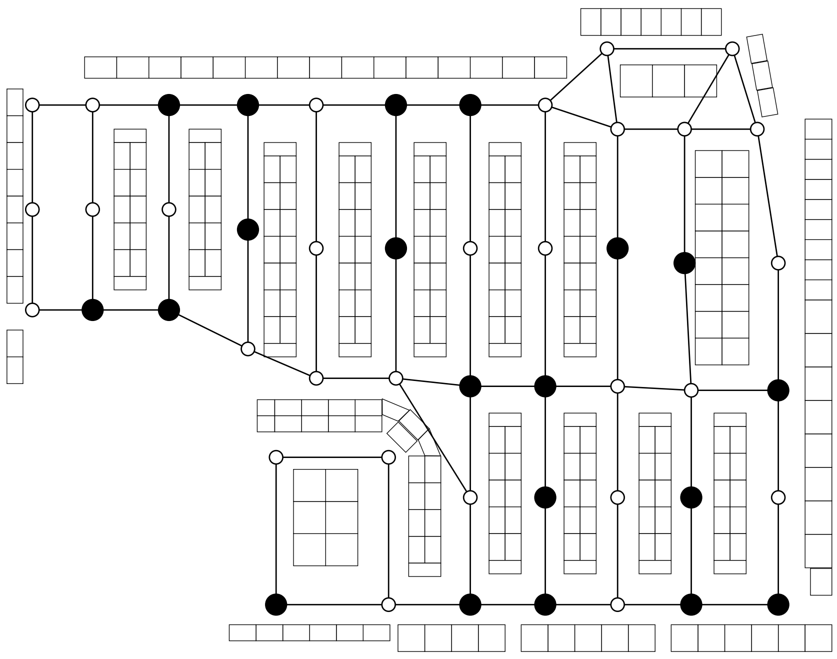


Figure 4.11: For the second experiment we randomly chose 20 starting locations (filled nodes) for the search trials. The target locations of the products remain the same as in the previous experiment (see Fig. 4.10).

Table 4.3: Mean and standard deviation (SD) of the overall search path lengths for different search strategies. For further comparison we list the length of the optimal path and the path length ratio defined as the average path length of a strategy divided by the length of the optimal path.

Strategy	Search Path Length		Ratio	Samples
	Mean (km)	SD (km)		
Exploration	32.2	1.30	6.2	1000
Dec. Tree (a–d, pruned)	27.0	0.96	5.2	1000
MaxEnt	17.4	–	3.3	–
Optimal Path	5.2	–	1.0	–

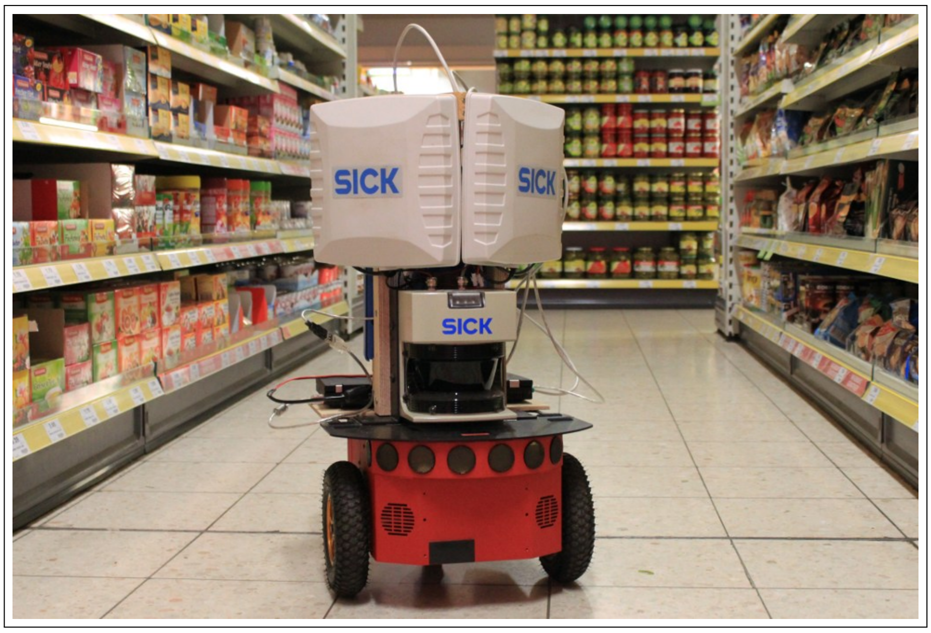


Figure 4.12: We equipped a Pioneer 3DX with a SICK laser range scanner and a SICK RFID reader with two antennas. The robot autonomously searched for a product in a supermarket using our proposed search strategy.

a significant improvement over the decision tree strategy, which is now more pronounced than in the previous experiment. To summarize, this second experiment confirms our finding of the first experiment, that it is beneficial to integrate domain-specific background knowledge when searching for an object.

4.3.3 Reactive Search Strategy – Searching with a Real Robot

As a proof of concept for the reactive search strategy, we let a mobile robot autonomously search for a product in a real supermarket. We used a Pioneer 3DX equipped with a SICK LMS 291 laser range scanner and a SICK RFI 641 RFID reader (Fig. 4.12). The target product was marked with a passive UHF RFID tag and the robot stopped searching as soon as it had detected the corresponding RFID tag of the product. For navigation purposes the robot mapped its environment in a local occupancy grid-map with a side length of 16 meters. The local map was successively re-centered at the robots location if the robot moved more than one meter. We used a virtual sensor for detecting the relevant edge attributes and the location of the decision points in the reference frame of the local map. Extracting this information directly from sensor data is a problem in its own right that we consider to be beyond the scope of this work which focuses on high-level decision making.

In the first experiment the robot started at the entrance of the market and had to find yogurt by utilizing the decision tree depicted in Fig. 4.4 (a) on page 82. In Fig. 4.13 we depict a sequence of snapshots[3] of this search run. The small image in the upper right corner shows the path taken by the robot as well as the current location of the robot and the local map with respect to a map of the whole market. The rest of the image shows a detail of the map with the current decision point and the possible successor nodes with their respective edges. The successor nodes of which the robot may choose randomly are marked by a black dot. These nodes either belong to edges that were classified as promising or to unvisited edges if no edge was classified as promising. As can be seen in Fig. 4.13 (first and second picture) the robot first proceeded straight down the main aisle, because at each decision point there was only one promising edge in front of the robot: an unvisited edge in a main aisle with a cooling shelf visible in its direction. At the end of the main aisle the robot then selected the only unvisited edge, which led to the node to the robot's left side. After a few more meters it finally detected the product's RFID tag and successfully ended the search (Fig. 4.13, third picture).

In the second experiment the robot started in a side aisle located nearly in the center of the market. At the beginning, it had two promising choices for leaving the side aisle and entering a main aisle and randomly chose to enter the lower main aisle. Arriving there (Fig. 4.14, first picture) it encountered only one promising direction: a node to its left side, which lies in a main aisle with a cooling shelf visible at its end. It then proceeded down the main aisle – encountering two similar situations – until it was left with a choice of two non-promising but unvisited edges at the end of the main aisle (Fig. 4.14, second picture). It randomly chose to turn right and successfully ended the search after detecting the RFID tag of the product after a few meters (Fig. 4.14, third picture).

As in the previous experiment, the robot made an optimal decision at each decision point. Of course, this will not always be the case. For example, in a replication of the second experiment the robot chose to proceed to the upper node when it once again was confronted with the situation depicted in the second picture of Fig. 4.14 in which it had to choose randomly among two nodes. This resulted in the longer search path shown in Fig. 4.15. In general, it is inevitable that the robot makes a suboptimal decision at some point during the search. The purpose of our proposed technique is to learn heuristics that

[3]Note that some parts of the real market have been rearranged while we have been working on this technique. This accounts for the differences in the market layout of Figs. 4.13–4.15 when compared to the previously shown figures in this chapter.

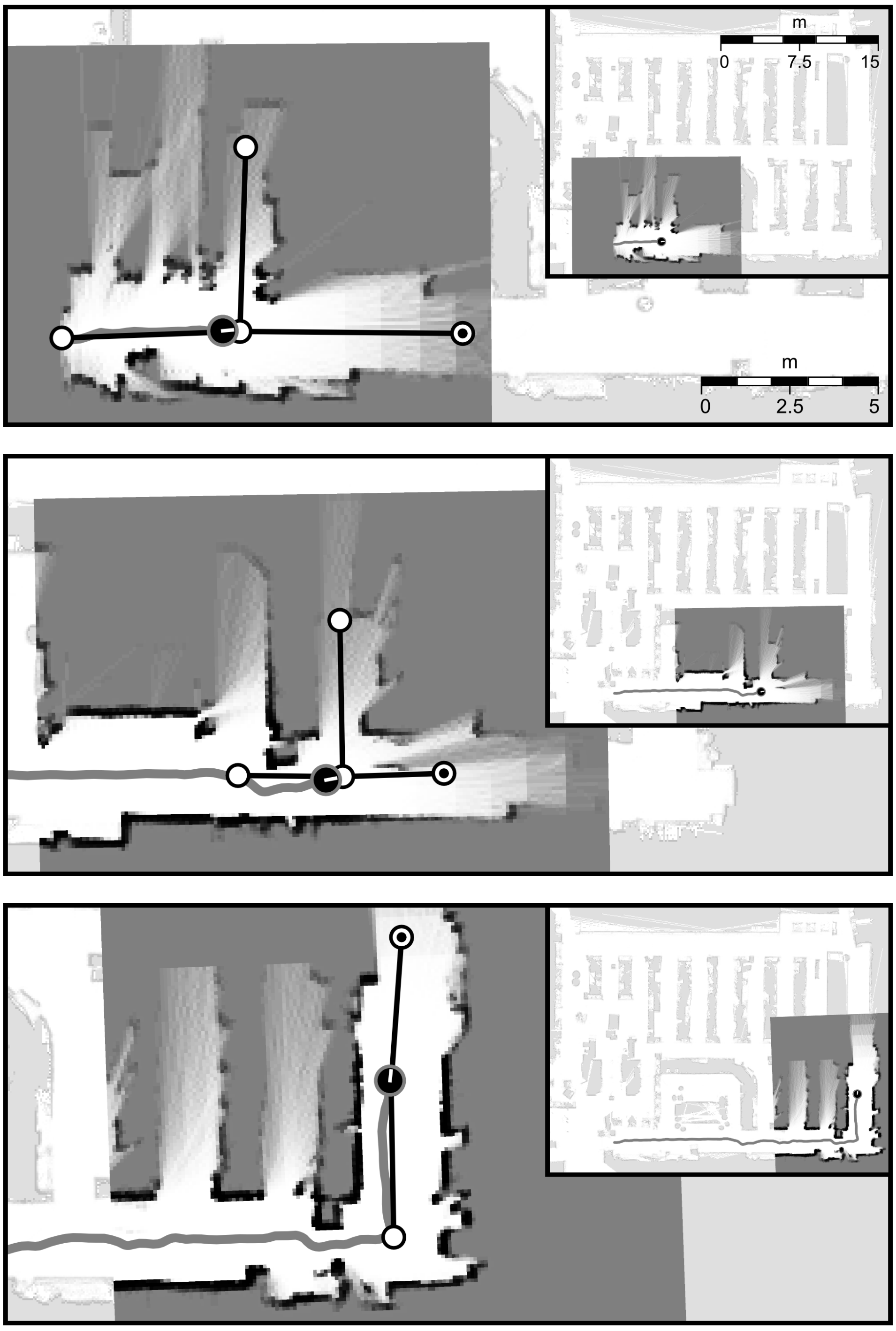


Figure 4.13: In the first search run the robot started at the entrance of the market.

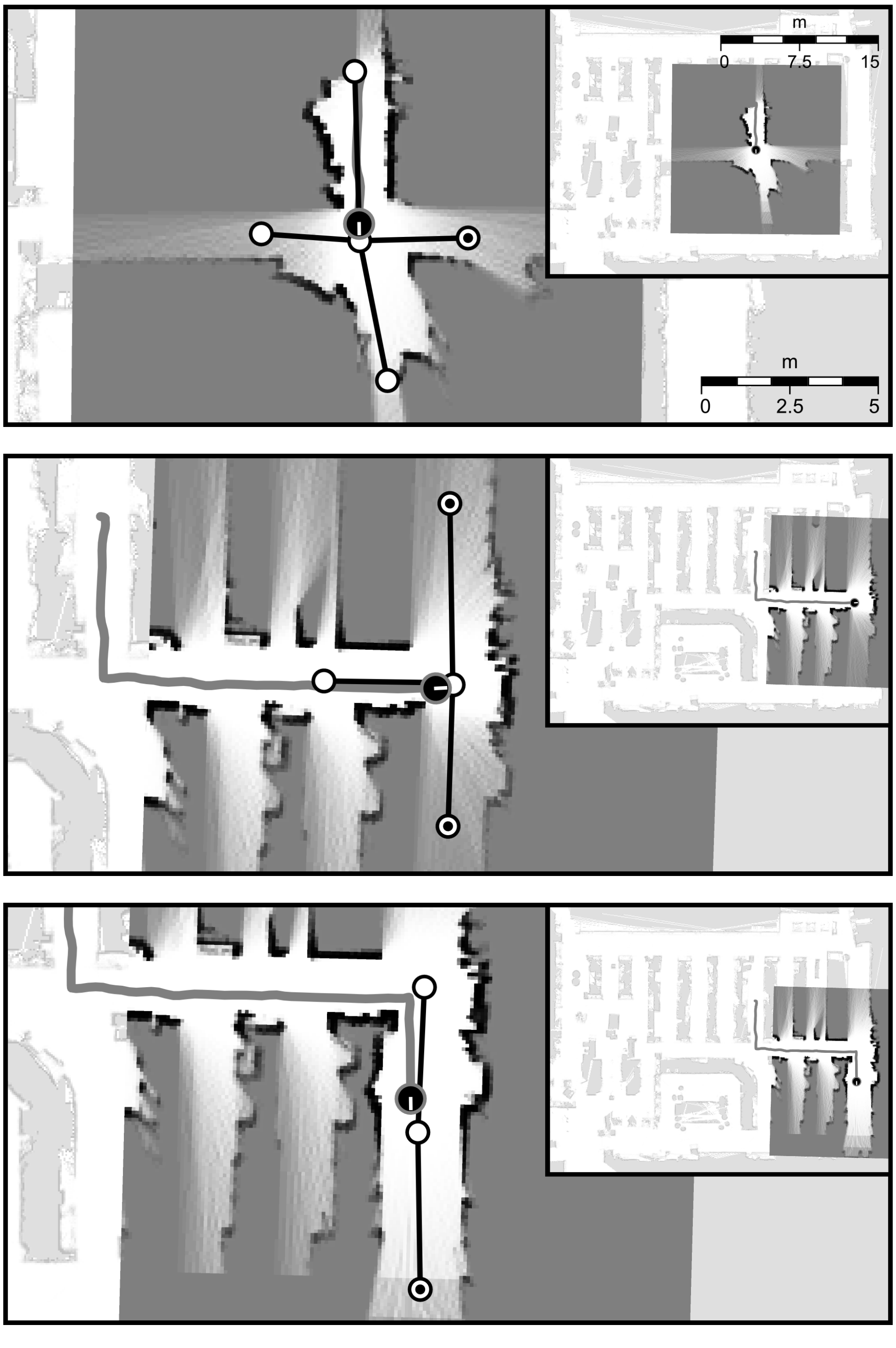


Figure 4.14: The next search run began in a side aisle in the center of the market.

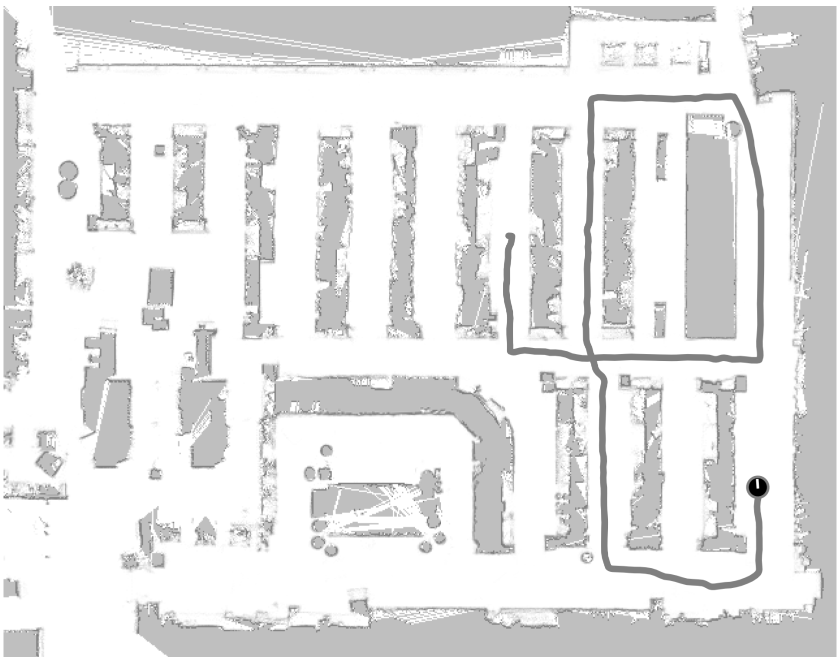


Figure 4.15: In a replication of the second search run the robot chose the upper node in the situation depicted in the second picture of Fig. 4.14, which resulted in a longer search path.

support the robot in making the right decision such that on average the product is found faster than with an uninformed search strategy. We believe that the results obtained both in simulation and real-world experiments highlight the potential of this idea.

4.3.4 Inference-based Search Strategy – Further Evaluations

For further evaluation of the inference-based search strategy, we search for all 141 products that are available in all four markets. Each product has to be searched for in each of the four markets. When searching in one market, the conditional distributions that capture our background knowledge were estimated from the remaining three markets. Thus, the market is completely unknown to the robot, although the robot has some background knowledge about the usual object arrangements stemming from the other three markets. The search starts at the entrance and ends at the node where the product is reachable, i.e. at the access node of the corresponding shelf. When searching for the next product the robot starts again at the entrance without knowledge of the market from the previous run.

As a performance measure we use the sum of the path lengths of all 4×141 individual search runs. The shortest path from the entrance to a product equals on average about

Table 4.4: Overall search path lengths for different search strategies.

Strategy	Path Length (10^3 m)	Overhead (to best)	Ratio (to short. path)
shortest path	13.4	$-57.6\,\%$	1.0
maxent (log.)	31.6	$0\,\%$	2.36
maxent (cont.)	32.3	$2.2\,\%$	2.41
maxent (discr.)	32.5	$2.9\,\%$	2.43
geom. mean	35.9	$13.6\,\%$	2.68
weight. avg.	36.6	$15.8\,\%$	2.73
geom. mean (mod. 1–3)	38.8	$22.8\,\%$	2.90
weight. avg. (mod. 1–3)	39.4	$24.7\,\%$	2.94
model 3	39.9	$26.3\,\%$	2.98
model 2	40.8	$29.1\,\%$	3.04
model 1	51.0	$61.4\,\%$	3.81
exploration	61.3 ($\sigma = 1.4$)	$94.0\,\%$	4.57

24 meters. The results are listed in Tab. 4.4, along with the path length ratio defined as the search path length divided by the shortest path. Additionally, we list the overhead relative to the best search strategy. As the exploration strategy chooses randomly among the directions at junctions, we re-evaluate it ten times and list the mean and standard deviation of the total path lengths. The other strategies are deterministic. We also evaluate the linear and the logarithmic opinion pool with uniform and normalized weights (we will thus refer to them as the weighted average and the geometric mean). Both fusion approaches are also evaluated using either only three local models p_{A_i,r_j} or just one model (in this case both fusion approaches are equivalent). These three local models can be considered to be the most specific ones: all use the "Category (fine)" attribute in combination with one of the following spatial relations: "same unit" (model 1), "adjacent unit" (model 2), or "shared path node unit" (model 3).

In general, the maxent search strategy achieves the shortest search path which is only about half as long as the one of the exploration strategy and about 2.4 times longer than the shortest possible path. The usage of different feature representations in the maxent model has only a mild influence on the resulting path lengths. Tab. 4.4 also highlights that the best performance is achieved by strategies that utilize all local models. In Fig. 4.16 we additionally plot the percentage of found products versus the path length ratio. In Fig. 4.17 we plot the average path lengths to a product when searching for multiple products. For

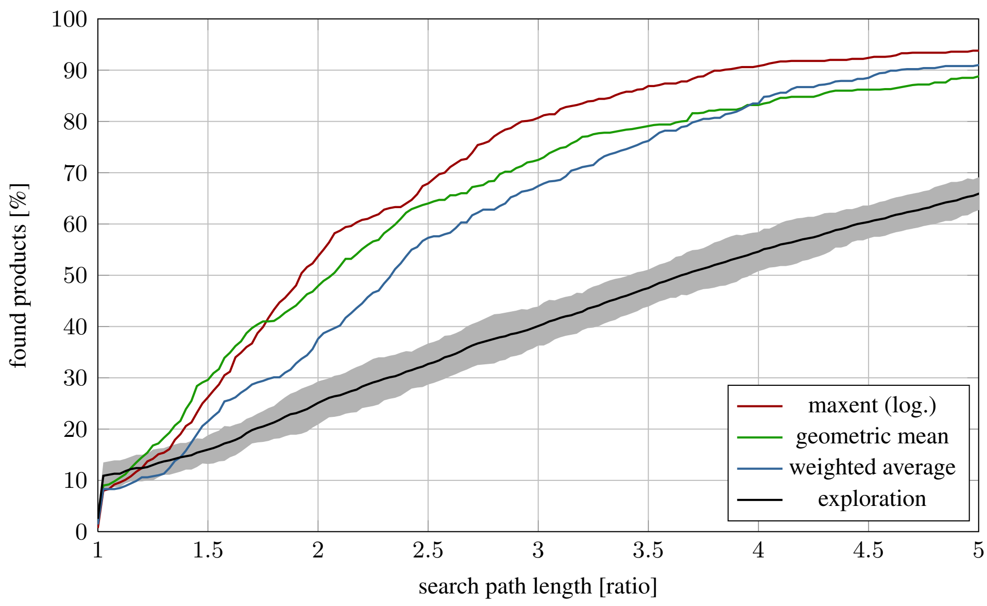


Figure 4.16: Searching for a single product.

this, we computed distributions for each of the query products and replaced the detection probability needed for computing the node utilities by the probability of finding *any* of the query products. The products may be found in any order. As can be seen, employing background knowledge also helps in this situation, though the benefits diminish as the number of objects increase, which is an expected result: the more objects the robot has to find, the more of the search area has to be visited anyway and a exploration strategy might ultimately perform equally well.

4.4 Related Work

There exists theoretical work on general search problems in the fields of robotics and artificial intelligence [38] as well as operations research [5]. Finding an optimal search path in a graph that either minimizes the expected time to detection [77] or the expected search costs [81] is known to be NP-hard. Besides complexity considerations in theoretical work, some prior work evaluated proposed search strategies in simulation. The approach presented in [41], for example, used a computationally involved dynamic programming technique for planning an optimal search path to find multiple stationary targets. In [9], a framework was proposed that additionally allows to reason about the possible absence of

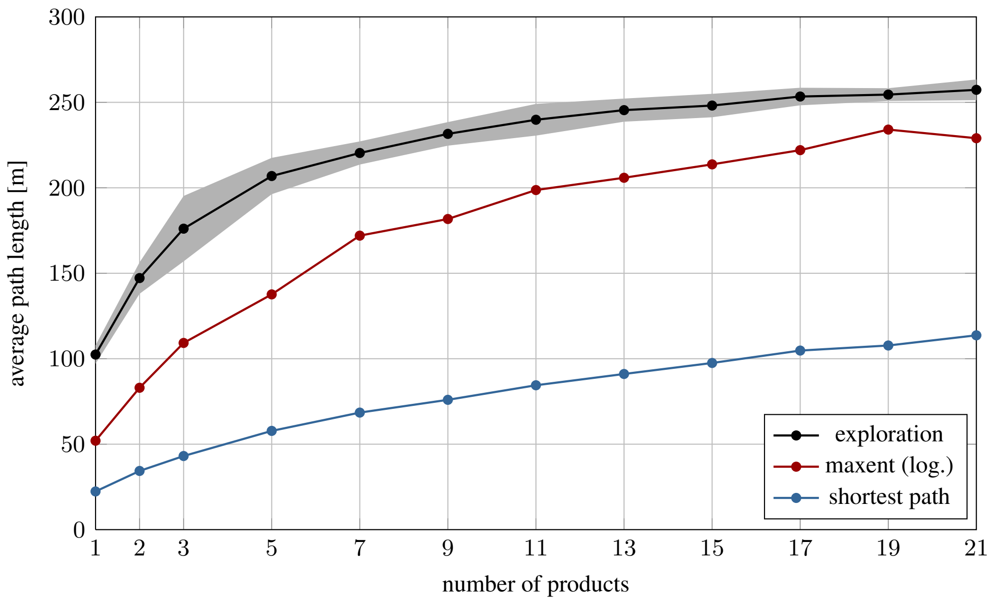


Figure 4.17: Searching for multiple products.

the target in the search area. In contrast to these works, we additionally assume that the environment is initially unknown. Most of these approaches allow to incorporate background knowledge as a prior distribution over the target location. But this distribution is assumed to be given in advance and then updated during the search based on simple presence or absence detections. Our work additionally aims at modeling such a distribution based on object co-occurrences, or by implicitly incorporating background knowledge as search heuristics.

Thus, a further related problem is how to model expectations about object arrangements in indoor environments. In prior work this has been realized by modeling different types of places in terms of object counts and inter-object distances [79], by utilizing object co-occurrence statistics [39], by utilizing a full 3D constellation model [53], or by using a manually designed ontology about indoor environments [83, 22]. Of these works, only [39], and to some extent [22], also considered the search problem and used their model for improving search efficiency.

Modeling background knowledge about indoor environments for improving search efficiency has received far less attention in the literature so far. In [39], known object locations in a given global map were used for efficiently finding another object at an unknown location. A Markov random field based on statistics of object co-occurrences was used to infer a likelihood map and to plan a search path that minimizes the expected search path length. Again, this is a work that assumed that the structure of the environment, and some of the objects therein, are known. In [10], the application scenario was to efficiently find the entrance hall in a hotel. A relational navigation policy was learned which utilized information about the type of rooms and corridors that are directly connected to the robot's current location. The work presented in [22] aimed at improving the task planning of a mobile robot by relying on semantic information about its domain. In particular, they defined an ontology about typical home-like environments and generated plans to find unseen objects or type of rooms, e.g. a bedroom.

4.5 Conclusions

We presented two approaches for efficiently finding an object in an unknown environment. The first approach is a reactive search technique that only depends on local information about the objects in the robot's vicinity when deciding where to search next. This strategy is based on search heuristics that can be learned from data of optimal search paths. As a proof of concept, we presented real-world experiments in which a mobile robot searched autonomously for a product using the proposed decision tree strategy. We furthermore presented a second, inference-based search strategy, that explicitly reasons about the location of the target object given all observations made so far. It thereby takes more global information into account. The underlying model of this MaxEnt search strategy can be learned from object arrangements of similarly structured example environments.

The MaxEnt strategy achieved shorter search paths than the decision tree strategy. But this advantage comes at additional cost as it needs to maintain and update a global map and perform inference by taking into account all previously seen products. This has important practical ramifications. The decision tree strategy can be implemented by using a local grid-map as we have demonstrated in the real-world experiments. Thus, the mapping overhead is constant in the size of the search area – the only exception being the decision point graph, which is needed for backtracking to a node with an unvisited edge if all outgoing edges of the current node have been visited already. However, this graph only needs to be topologically correct and the computational overhead during the search can be considered negligible compared to the maintenance of the local grid-map. The MaxEnt strategy, on the other hand, needs to resort to a computationally much more demanding online SLAM approach for maintaining a consistent global map that preserves the spatial relations between all detected objects. On the downside the mapping overhead now grows linearly with the size of the search area (or linearly with the length of the path the robot has taken). The upside is that more informed decisions can be made which eventually leads to shorter search paths.

To summarize, in this chapter we presented two general approaches for modeling, learning, and utilizing background knowledge about indoor environments such that a mobile robot is able to find an object in an initially unknown environment more efficiently than would have been possible without such domain-specific knowledge. The choice between these two approaches constitutes a trade-off between the complexity of the underlying model and the resulting search efficiency. Extensive experiments showed that both strategies significantly outperform an exploration strategy. This demonstrates the benefits of utilizing background knowledge when searching for objects in unknown environments.

The inference-based search technique relied on object co-occurrence statistics. The parameters of the model can be learned based on the layout of supermarkets by generating statistics about how often certain objects co-occur within a given spatial context. However, these spatial contexts are induced by the local models which use manually defined spatial relations – like objects being “in the same aisle” or “in the same shelf”. This motivates our work in the next chapter, in which we wish to learn the relevant spatial relations between objects in an unsupervised way. Thereby, we hope to increase the autonomy and adaptivity of service robots by side-stepping the reliance on predefined spatial relations derived from domain-specific expert knowledge.

CHAPTER 5

Unsupervised Learning of Object Constellations

Robots operating in domestic environments need to deal with a variety of different objects. Often, these objects are neither placed randomly, nor independently of each other. For example, objects on a breakfast table such as plates, knives, or bowls typically occur in recurrent configurations. In this chapter, we propose a novel hierarchical generative model to reason about latent object constellations in a scene. The proposed model is a combination of a Dirichlet process and beta processes, which allows for a probabilistic treatment of the unknown dimensionality of the parameter space. We show how the model can be employed to address a set of different tasks in scene understanding including unsupervised scene segmentation and completion of partially specified scenes. We describe how to sample from the posterior distribution of the model using Markov chain Monte Carlo (MCMC) techniques and present an experimental evaluation with simulated as well as real-world data obtained with a Kinect camera.

Imagine a person laying a breakfast table and the person gets interrupted so that she cannot continue with the breakfast preparation. A service robot, such as the one depicted in Fig. 5.1 on the next page, should be able to proceed laying the table without receiving specific instructions. It faces a series of questions: how to infer the total number of covers, how to infer which objects are missing on the table, and how should the missing parts be

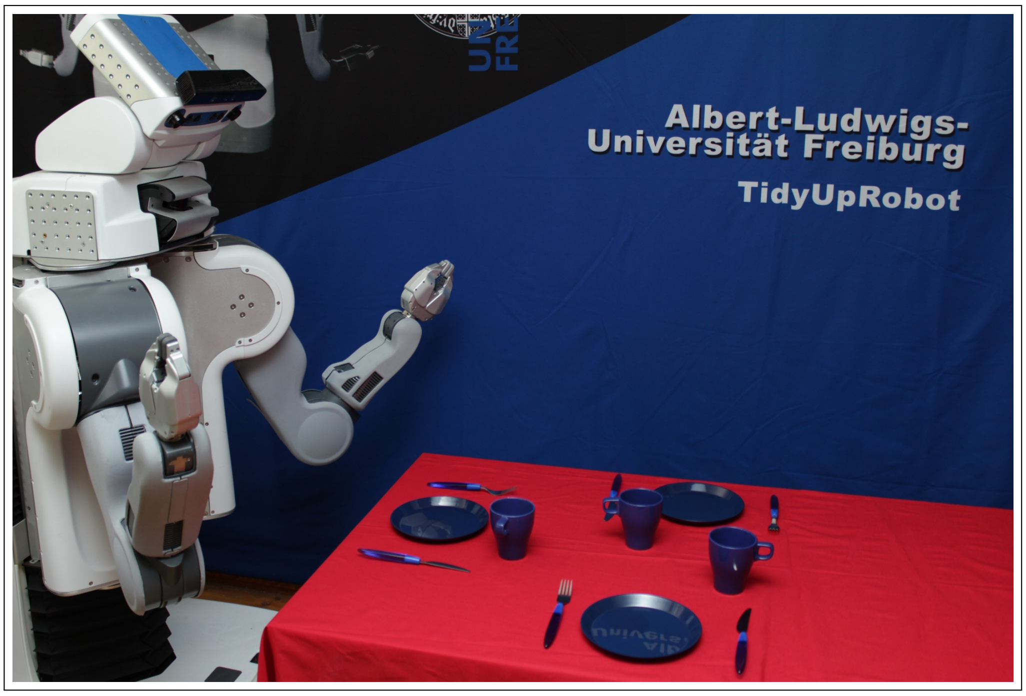
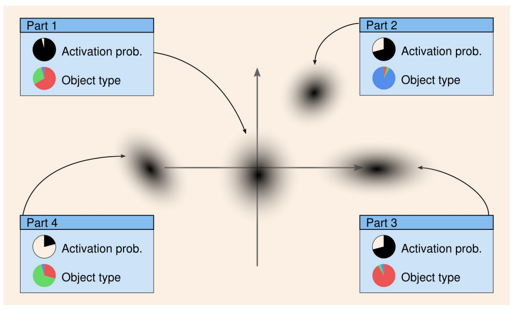


Figure 5.1: A scene typically contains several observable objects and the task is to infer the latent meta-objects where a meta-object is considered to be a constellation of observable objects. At a breakfast table, for example, the meta-objects might be the covers that consist of the observable objects plate, knife, fork, and cup.

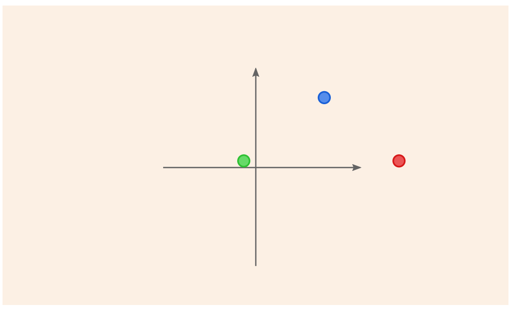
arranged? For this, the robot should not require any user-specific pre-programmed model but should ground its decision based on the breakfast tables it has seen in the past.

In this chapter, we address the problem of scene understanding given a set of unlabeled examples and generating a plausible configuration from a partially specified scene. The *key contribution* of our technique is the definition of a novel hierarchical nonparametric Bayesian model to represent the scene structure in terms of object groups and their spatial configuration. We show how to infer the scene structure in an unsupervised fashion by using Markov chain Monte Carlo (MCMC) techniques to sample from the posterior distribution of the latent scene structure given the observed objects.

In our model, each scene contains an unknown number of latent object constellations or *meta-object instances*. In the breakfast table example, a place cover can be seen as a meta-object instance of a certain type that, for example, consists of the objects plate, knife, and cup. An instance of a different type might consist of a cereal bowl and a spoon. These meta-object instances are assumed to be sampled from a distribution over object constellations.



(a) Meta-object type.



(b) Meta-object instance.

Figure 5.2: A meta-object *type* is modeled as a collection of parts, each having a Gaussian distribution, a multinomial distribution over object types, and a binary activation probability. A meta-object type is a distribution over object constellations, which we will call meta-object *instances*.

We will refer to these distributions as meta-object *types* (see Fig. 5.2) as we assume that there are different types of constellations. Thus, not all meta-object instances of a given type are the same, they differ in the sense that some objects may be missing and that the objects may not be arranged in the exact same way.

When specifying a generative model for our problem, we have the difficulty that the dimensionality of the model is part of the learning problem. This means, that besides learning the parameters of the model, like the pose of a meta-object, we additionally need to infer the *number* of involved meta-objects, meta-object parts, etc. The standard solution would be to follow the model selection approaches, for example, learning several models and then choosing the best one. Such a comparison is typically done by trading off the data likelihood with the model complexity as, for example, done for the Bayesian information criterion (BIC). The problem with this approach is the huge number of possible models, which renders this approach intractable in our case.

To avoid this complexity, we follow another approach, motivated by recent developments in the field of hierarchical nonparametric Bayesian models based on the Dirichlet process and the beta process. These models are able to adjust their complexity according to the given data, thereby sidestepping the need to select among several finite-dimensional model alternatives. Based on a prior over scenes, which is updated by observed training scenes, the model can be used for parsing new scenes or completing partially specified scenes by sampling the missing objects.

Whereas in this chapter, we consider the problem of learning the object constellations on a breakfast table as depicted in Fig. 5.1, our model is general and not restricted to this scenario.

5.1 Generative Scene Model

In this section, we describe the proposed generative scene model. We assume that the reader is familiar with the basics of nonparametric Bayesian models [23], especially with the Chinese restaurant process (CRP) and the Dirichlet process (DP) [70], the (two-parameter) Indian buffet process (IBP) and the beta process (BP) [24, 73], and the concepts of hierarchical [71] and nested [57] processes in this context.

In the following, we consider a scene as a collection of observable objects represented as labeled points in the 2D plane. The 2D assumption is due to our motivation to model table scenes. However, the model is not specifically geared towards 2D data and could in principle also be applied to 3D data. Basically, we assume that each scene contains an unknown number of latent object constellations (place covers). An object constellation is called a meta-object *instance* (or simply meta-object) and corresponds to a sample from a meta-object *type*, which is a distribution over object constellations and is represented as a part-based model with infinitely many parts. As illustrated in Fig. 5.2, each part has a binary activation probability, a Gaussian distribution over the relative object position, and a multinomial distribution over the object type (knife, fork, etc.). To sample from a meta-object type, one first samples the activation of each part. For each activated part, one then samples the relative position and the object type to be generated at this location. Each activated part generates exactly one object per meta-object instance. Thus, the objects of a scene can be grouped into clusters. Each cluster corresponds to a meta-object instance and the objects of a cluster can be associated to the parts of the corresponding meta-object type. Please note, that we will use the terms "meta-object" and "cover" interchangeably.

5.1.1 Description of the Generative Process

Following the example given in the introduction, imagine that our robot's goal is to set a table for a typical family breakfast. At the beginning, it enters a room with an empty breakfast table and an infinite number of side tables, each holding a prototypical cover. It estimates the area A of the breakfast table surface and boldly decides that $n \sim \mathrm{Pois}(A\lambda)$ covers are just right. The robot chooses one of the side tables and finds a note with the address of a Chinese restaurant where it can get the cover. Arriving there, it sees again infinitely many tables each corresponding to a particular cover type and each displaying a count of how often someone took a cover from this table. It is fine with just about any cover type and decides randomly based on the counts displayed at the tables, even considering a previously unvisited table. At that table there is another note redirecting to an Indian restaurant. In this restaurant, it is being told that the cover needs to be assembled by choosing the parts that make up this cover. The robot randomly selects the parts based on their popularity and even considers to use a few parts no one has ever used before. For each chosen part its relative position is sampled from the part's Gaussian distribution and then the robot samples the object from the part's multinomial distribution over types (plate, knife, fork, etc.). Having assembled the cover this way, the robot returns to the breakfast table and puts it randomly on it. The process is then repeated for the remaining $n - 1$ covers. See also Fig. 5.3 on the next page for an overview of the model structure in terms of the various CRPs and IBPs.

More formally, we have a hierarchical model with a high-level Dirichlet process DP_t, a low-level beta process BP_c and a further independent beta process BP_ϵ. First, we draw G_t from the high-level DP_t

$$G_t \sim \mathrm{DP}_t(\alpha_t, \mathrm{BP}_p(c_p, \alpha_p, \mathrm{Dir} \times \mathcal{NW})), \tag{5.1}$$

where the base distribution of DP_t is a beta process BP_p modeling the parts' parameters. This is done only once and all scenes to be generated will make use of the same draw G_t. This draw describes the distribution over all possible meta-object types (cover types) and corresponds to the Chinese restaurant mentioned above. The base distribution of the beta process is the prior distribution over the part parameters. The parameters are a 2D Gaussian distribution over the relative location and a multinomial over the observable object types. The parameters are sampled independently and we use their conjugate priors in the base distribution, i.e., a (symmetric) Dirichlet distribution Dir for the multinomial and the normal-Wishart distribution $\mathcal{NW}$ [45] for the 2D Gaussian. The mass parameter α_p of BP_p is our prior over the number of activated parts of a single meta-object instance and the concentration parameter c_p influences the total number of instantiated parts across all instances of the same type. Likewise, the parameter α_t influences the expected number of meta-object types. Each scene s has its own meta-object IBP and the meta-object instances are determined by a single draw from the corresponding beta-Bernoulli process as follows:

$$\begin{aligned}
G_c^{(s)} &\sim \mathrm{BP}_c(1, |A_s|\,\alpha_c,\, G_t \times U(A_s \times [-\pi, \pi])) && && (5.2)\\
\{G_{t_j}, T_j\}_j &\sim \mathrm{BeP}(G_c^{(s)}) && && (5.3)\\
\{\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k, \gamma_k\}_k &\sim \mathrm{BeP}(G_{t_j}) && \text{for each } j && (5.4)\\
\{\mathbf{x}, \omega\} &\sim p(\mathbf{z} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k, \gamma_k, T_j) && \text{for each } k && (5.5)
\end{aligned}$$

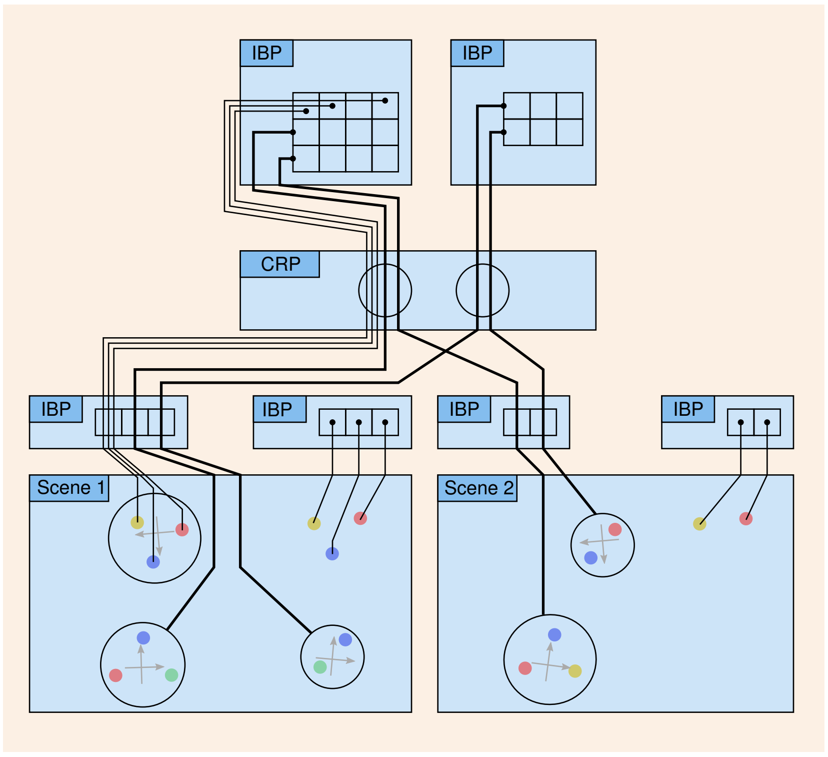


Figure 5.3: Basic structure of our model: The Indian buffet processes (IBP) on top are nested within the tables (represented as circles) of the Chinese restaurant process (CRP) below them. The blocks within the IBP frames represent the relevant part of the IBP matrix (see Fig. 5.4). Clusters in the scenes are meta-object instances and objects (colored points) of the clusters need to be associated to entries in the IBPs shown on top. This is explicitly shown for one cluster in the first scene. For visibility reasons, the rest of the associations are drawn as a single thick line. At the lowest level, each scene has a meta-object IBP (shown on the left) and a noise IBP (on the right) from which the meta-object instances and the noise objects of a scene are drawn.

In Eq. (5.2), the concentration parameter is irrelevant and arbitrarily set to one. The reason for this is, that the scene-specific IBPs only have a single customer which selects a Poisson distributed number of dishes (meta-objects). This distribution for the first customer is only influenced by the mass parameter and not the concentration parameter. While the concentration parameters is set to one, the mass parameter $|A_s|\, \alpha_c$ is the mean of the said Poisson distribution and therefore corresponds to the expected number of meta-object instances in a scene. Here, $|A_s|$ is the table area in scene s. Thus, the greater the table area, the more meta-object instances we expect a priori. The base distribution of BP_c in Eq. (5.2) samples the parameters of a instance j: its type t_j and its pose T_j. The type t_j is drawn from the distribution over meta-object types G_t from Eq. (5.1). The pose T_j is

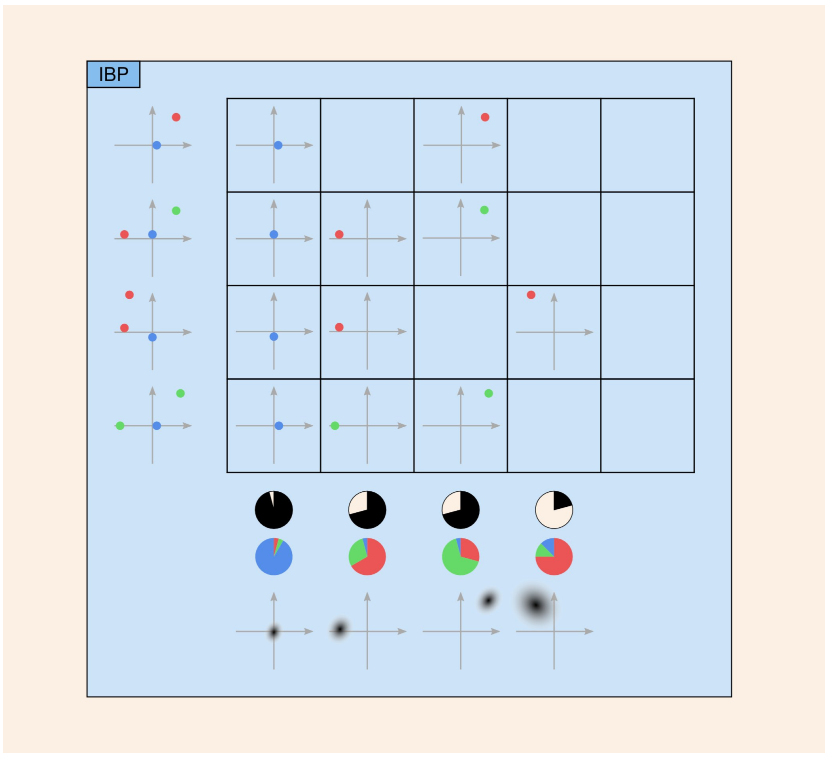


Figure 5.4: A more detailed view on a part IBP representing a meta-object type. This would correspond to one of the IBPs at the very top in Fig. 5.3. The rows represent customers and the columns represent dishes (parts). The customers of this IBP are the meta-object *instances* associated to this meta-object *type* in any of the scenes. The objects of the instances must be associated to one of the parts. They thereby update the posterior predictive distribution (illustrated at the bottom) over the objects of a new customer. Remember, that the actual part parameters are integrated out due to the usage of conjugate priors.

drawn from a uniform distribution U over the pose space $A_s \times [-\pi, \pi]$ where A_s is the table area. Each atom selected by the Bernoulli process in Eq. (5.3) corresponds to a meta-object instance and this selection process corresponds to the side table metaphor mentioned above. The meta-object type t_j basically references a draw G_{t_j} from the nested beta process in Eq. (5.1) which models the parts of a meta-object type. Thus, in Eq. (5.4) we need another draw from a Bernoulli process to sample the activated parts for this instance, which yields the Gaussians $(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ and the multinomials $(\boldsymbol{\gamma}_k)$ for each active part k. Finally, in Eq. (5.5) we draw the actual observable data from the data distribution as realizations from the multinomials and the (transformed) Gaussians, which yields an object $\mathbf{z} = \{\mathbf{x}, \omega\}$ on the table with location $\mathbf{x}$ and type ω for each activated part.

Each scene has an additional independent beta process BP_ϵ

$$G_\epsilon^{(s)} \sim \mathrm{BP}_\epsilon(1, \alpha_\epsilon, M \times U(A_s)) \tag{5.6}$$

$$\{\mathbf{x}_i, \omega_i\}_i \sim \mathrm{BeP}(G_\epsilon^{(s)}) \tag{5.7}$$

that directly samples objects (instead of meta-objects) at random locations in the scene. Here, M is a multinomial over the observable object types and $U(A_s)$ is the uniform distribution over the table area A_s. This beta-Bernoulli process will mainly serve as a "noise model" during MCMC inference to account for yet unexplained objects in the scene. Accordingly, we set the parameter α_ϵ to a rather low value to penalize scenes with many unexplained objects. Figs. 5.3 and 5.4 illustrate the overall structure of the model.

5.1.2 Posterior Inference in the Model

Some model parameters are integrated out analytically. This holds true for the draws from the various processes, which means that we are only working in terms of posterior predictive distributions as modeled by the Chinese restaurant and Indian buffet representations. Further, some of the atoms (table and dish parameters) are also integrated out. For example, the dish parameters of the part IBPs consist of Gaussians and multinomials, which can be integrated out analytically due to the usage of conjugate prior. What remains to be explicitly represented are the poses and types of the meta-object instances and the associations of the observable objects to either the noise process or to a part of a meta-object instance.

We now describe how to sample from the posterior distribution over the latent variables $\{\mathbf{C}, \mathbf{a}\}$ given the observations $\mathbf{z}$. We use the following notation: The observations are the objects of all scenes and a single object $\mathbf{z}_i = \{\mathbf{x}_i, \omega_i\}$ has a 2D location $\mathbf{x}_i$ on the table and a discrete object type ω_i. A meta-object instance j has parameters $C_j = \{T_j, t_j, \mathbf{d}_j\}$, where T_j denotes the pose and t_j denotes the meta-object type, which is an index to a table in the type CRP, and $\mathbf{d}_j$ denotes part activations and associations to the observable objects. If $d_{j,k} = 0$ then part k of meta-object instance j is inactive, where k is an index to a dish in the corresponding part IBP (which is nested in the CRP table t_j). Otherwise, the part is active and $d_{j,k} \neq 0$ is a reference to the associated observable object, i.e., $d_{j,k} = i$ if it generated the object $\mathbf{z}_i$. Next, for each scene we have the associations $\mathbf{a}$ to the noise IBP, i.e., the list of objects currently not associated to any meta-object instance. For ease of notation, we will use index functions $[\cdot]$ in an intuitive way, for example, $\mathbf{t}_{[-j]}$ denotes all type assignments except the type assignment t_j of meta-object j, and $\mathbf{T}_{[t_j]}$ denotes all poses of meta-objects that have the same type t_j as meta-object j, and $\mathbf{z}_{[\mathbf{d}]}$ are all observations referenced by the associations $\mathbf{d}$, etc.

We employ Metropolis-Hastings (MH) moves to sample from the posterior [2], which allows for big steps in the state space by updating several strongly correlated variables at a time. In the starting state of the Markov chain, all objects are assigned to the noise IBP of their respective scene and thus are interpreted as yet unexplained objects. We sample for a fixed yet sufficiently high number of iterations to be sure that the Markov chain converged. We use several types of MH moves, which we will explain in detail after describing the joint distribution.

5.1.2.1 Joint Distribution

The joint distribution $p(\mathbf{C}, \mathbf{a}, \mathbf{z}) = p(\mathbf{T}, \mathbf{t}, \mathbf{d}, \mathbf{a}, \mathbf{z})$ is defined in terms of the CRP and IBP representation where the draws from the processes have been integrated out. It includes the poses $\mathbf{T}$ of all meta-object instances, their types $\mathbf{t}$ (table indices of the type CRP), their active parts $\mathbf{d}$ (dish indices of the corresponding part IBP), the associations $\mathbf{a}$ of objects to the noise process, and the observed data $\mathbf{z}$.

$$\begin{aligned} p(\mathbf{C}, \mathbf{a}, \mathbf{z}) &= p(\mathbf{T}, \mathbf{t}, \mathbf{d}, \mathbf{a}, \mathbf{z}) && (5.8) \\ &= \left(\prod_{s=1}^{S} p(n_{s,\epsilon}) p(n_{s,m}) \right) p(\mathbf{t}) \left(\prod_{t=1}^{n_t} p(\mathbf{d}_{[t]} \mid \mathbf{t}) \right) \\ &\quad p(\mathbf{T}) \left(\prod_{t=1}^{n_t} \prod_{k=1}^{K_t} p(\mathbf{z}_{[\mathbf{d}_{[t,k]}]} \mid \mathbf{T}_{[t]}, \mathbf{d}_{[t,k]}, \mathbf{t}) \right) && (5.9) \end{aligned}$$

Here, $n_{s,m}$ is the number of meta-object instances and $n_{s,\epsilon}$ is the number of noise objects in scene s. Each dish parameter of the noise IBP directly corresponds to the parameters $\mathbf{z}_i = \{\mathbf{x}_i, \omega_i\}$ of the associated noise object, as we assumed that there is no data distribution associated with these dishes. The base distribution for the dish parameters consists of independent and uniform priors over the table area and the object types, and so each dish parameter has the likelihood $(n_\omega \, |A_s|)^{-1}$, where n_ω is the number of observable object types and $|A_s|$ is the area of the table in scene s. Thus, the probability of the objects $\mathbf{z}_{[\mathbf{a}_{[s]}]}$ associated to a scene's noise IBP only depends on the *number* of noise objects and not on the particular type or position of these objects. However, it would be straightforward to use a non-uniform base distribution. Denoting the Poisson distribution with mean λ as

$\mathrm{Pois}(\cdot \mid \lambda)$ we thus have

$$p(n_{s,\epsilon}) = p(\mathbf{z}_{[\mathbf{a}_{[s]}]}, \mathbf{a}_{[s]}) = n_{s,\epsilon}!\,\mathrm{Pois}(n_{s,\epsilon} \mid \alpha_\epsilon)(n_\omega\,|A_s|)^{-n_{s,\epsilon}}. \tag{5.10}$$

Next, $p(n_{s,m}) = n_{s,m}!\,\mathrm{Pois}(n_{s,m} \mid \alpha_m\,|A_s|)$ is the prior probability for having $n_{s,m}$ meta-objects in scene s. The dish parameters of a meta-object IBP are the meta-object parameters C_j, consisting of the pose, type, and part activations. The likelihood for sampling a pose $p(T_j) = (|A_s|\,2\pi)^{-1}$ is uniform over the table surface and uniform in orientation, hence $p(\mathbf{T}) = \prod_{s=1}^{S}(|A_s|\,2\pi)^{-n_{s,m}}$. Next, $p(\mathbf{t})$ is the CRP prior for the meta-object types of all meta-object instances. The factors $p(\mathbf{d}_{[t]} \mid \mathbf{t})$ are the IBP priors for the part activations for all meta-object instances of type t and there are n_t different types currently instantiated. During MCMC sampling, we will only need the conditional for a single meta-object j

$$p(t_j, \mathbf{d}_j \mid \mathbf{t}_{[-j]}, \mathbf{d}_{[-j]}) = p(\mathbf{d}_j \mid \mathbf{d}_{[-j,t_j]}, \mathbf{t})p(t_j \mid \mathbf{t}_{[-j]}). \tag{5.11}$$

Here, $p(t_j \mid \mathbf{t}_{[-j]})$ is the CRP predictive distribution

$$p(t_j = i \mid \mathbf{t}_{[-j]}) = \begin{cases} \dfrac{n_i}{\alpha_c + \sum_{i'} n_{i'}} & i \text{ is an existing type} \\ \dfrac{\alpha_c}{\alpha_c + \sum_{i'} n_{i'}} & i \text{ is a new type} \end{cases}, \tag{5.12}$$

and n_i is the number of meta-object instances of type i (not counting instance j) and α_c is the concentration parameter of the CRP. Further, $p(\mathbf{d}_j \mid \mathbf{d}_{[-j,t_j]}, \mathbf{t})$ is the predictive distribution of the two-parameter IBP, which factors into activation probabilities for each of the existing parts and an additional factor for the number of new parts, denoted as n_+. An existing part is a part that has been activated by at least one other meta-object instance of this type in any of the scenes. The activation probability for an existing part k is

$$p(d_{j,k} \neq 0 \mid \mathbf{d}_{[-j,t_j]}, \mathbf{t}) = \frac{n_k}{n_{t_j} + c_p}, \tag{5.13}$$

where c_p is the concentration parameter of the part IBP, n_k is the number of meta-object instances that have part k activated in any of the scenes, and n_{t_j} is the total number of meta-object instances of type t_j in all scenes (the counts exclude the meta-object j itself). The probability for having n_+ associations to new parts is

$$p(\mathbf{d}_{[j,+]} \mid \mathbf{d}_{[-j,t_j]}, \mathbf{t}) = n_+!\,\mathrm{Pois}\left(n_+ \middle| \frac{c_p \alpha_p}{n_{t_j} + c_p}\right), \tag{5.14}$$

where α_p is the mass parameter of the part IBP, and $\mathbf{d}_{[j,+]}$ denotes the associations to new parts.

As stated in Eq. (5.8), the data likelihood $p(\mathbf{z}_{[\mathbf{d}]} \mid \mathbf{T}, \mathbf{d}, \mathbf{t})$ for the objects associated to meta-objects factors into likelihoods for each individual part k. Further, it factors into a spatial component and a component for the observable object type. As the meta-object poses $\mathbf{T}$ are given, we can transform the absolute positions $\mathbf{x}_{[\mathbf{d}_{[t,k]}]}$ of the objects associated to a certain part k of meta-object type t into relative positions $\tilde{\mathbf{x}}_{[\mathbf{d}_{[t,k]}]}$ with respect to a common meta-object reference frame. The relative positions are assumed to be sampled from the part's Gaussian distribution which in turn is sampled from a normal-Wishart distribution. As the Gaussian and the normal-Wishart distribution form a conjugate pair, we can analytically integrate out the part's Gaussian distribution which therefore does not have to be explicitly represented. Hence, the joint likelihood for $\tilde{\mathbf{x}}_{[\mathbf{d}_{[t,k]}]}$ of part k is computed as the marginal likelihood under a normal-Wishart prior. During MCMC inference, we only need to work with the posterior predictive distribution

$$p(\tilde{\mathbf{x}}_{[d_{j,k}]} \mid \tilde{\mathbf{x}}_{[\mathbf{d}_{[-j,t_j,k]}]}) = t_\nu(\tilde{\mathbf{x}}_{[d_{j,k}]} \mid \mu, \Sigma) \tag{5.15}$$

for a single relative position given the rest. This is a multivariate t-distribution t_ν with parameters μ, Σ depending both on $\tilde{\mathbf{x}}_{[\mathbf{d}_{[-j,t_j,k]}]}$ and the parameters of the normal-Wishart prior – for details see [45] and Section A.1 in the appendix. Similarly, the part's multinomial distribution over the observable object types can be integrated out as it forms a conjugate pair with the Dirichlet distribution. The posterior predictive distribution for a single object type is

$$p(\omega_{[d_{j,k}]} \mid \omega_{[\mathbf{d}_{[-j,t_j,k]}]}) = \frac{n_\omega + \alpha_\omega}{\sum_{\omega'}(n_{\omega'} + \alpha_{\omega'})}, \tag{5.16}$$

where n_ω is the number of times an object of type ω has been associated to part k of this meta-object type t_j, and α_ω is the pseudo-count of the Dirichlet prior. When describing the MCMC moves, we will sometimes make use of the predictive likelihood for all objects $\mathbf{z}_{[\mathbf{d}_j]}$ associated to a single meta-object instance j. This likelihood factors into the posterior predictive distributions of the individual parts and their spatial and object type components, as described above. In the following, we will describe the various MCMC moves in detail.

5.1.2.2 Death (Birth) Move $(T_j, t_j, \mathbf{d}_j, \mathbf{a}) \to (\mathbf{a}^\star)$

A death move selects a meta-object j uniformly at random, adds all of its currently associated objects to the noise process and removes the meta-object j from the model. The proposal probability for this move is $q_d(\mathbf{C}_{-j}, \mathbf{a}^\star \mid C_j, \mathbf{C}_{-j}, \mathbf{a}) = (n_m)^{-1}$ where n_m denotes the number of instantiated meta-objects in this scene before the death move. To simplify notation, we will just write $q_d(C_j)$ for the probability of deleting meta-object j. The reverse proposal is the birth proposal $q_b(C_j^\star, \mathbf{C}_{-j}, \mathbf{a}^\star \mid \mathbf{C}_{-j}, \mathbf{a}, \mathbf{z})$ that proposes new parameters $C_j^\star = \{T_j^\star, t_j^\star, \mathbf{d}_j^\star\}$ for an additional meta-object: the pose $T_j^\star$, the type $t_j^\star$, and the associations $\mathbf{d}_j^\star$. The new meta-object may reference any of the objects previously associated to the noise process and any non-referenced noise objects remain associated to the noise process. We will describe the details of the birth proposal in detail later on. To simplify notation, we will just write $q_b(C_j)$ for the birth proposal. Plugging in the model and proposal distributions in the MH ratio and simplifying we arrive at the acceptance ratio of the death move

$$R_d = \frac{1}{p(\mathbf{z}_{[\mathbf{d}_j]} \mid \mathbf{z}_{[\mathbf{d}_{[-j,t_j]}]}, \mathbf{T}_{[t_j]}, \mathbf{d}_{[t_j]}, \mathbf{t})p(t_j, \mathbf{d}_j \mid \mathbf{t}_{[-j]}, \mathbf{d}_{[-j]})} \frac{1}{p(T_j)} \frac{p(n_m - 1)}{p(n_m)} \frac{p(n_\epsilon + n_j)}{p(n_\epsilon)} \frac{q_b(C_j)}{q_d(C_j)}. \tag{5.17}$$

The counts n_j, n_m, and n_ϵ refer to the state before the death move, and n_m denotes the number of meta-objects in this scene, n_j are the number of objects currently associated to meta-object j, and n_ϵ is the number of noise objects. The ratio of a birth move is derived similarly.

5.1.2.3 Switch Move $(T_j, t_j, \mathbf{d}_j, \mathbf{a}) \to (T_j^\star, t_j^\star, \mathbf{d}_j^\star, \mathbf{a}^\star)$

This move is a combined death and birth move. It removes a meta-object and then proposes a new meta-object using the birth proposal. Thus, the number of meta-objects remains the same but one meta-object simultaneously changes its type t_j, pose T_j, and part associations $\mathbf{d}_j$. The death proposals cancel out and the acceptance ratio of this move is

$$R_s = \frac{p(\mathbf{z}_{[\mathbf{d}_j^\star]} \mid \mathbf{z}_{[\mathbf{d}_{[-j,t_j^\star]}]}, T_j^\star, \mathbf{T}_{[-j,t_j^\star]}, \mathbf{d}_j^\star, \mathbf{d}_{[-j,t_j^\star]}, t_j^\star, \mathbf{t}_{[-j]})}{p(\mathbf{z}_{[\mathbf{d}_j]} \mid \mathbf{z}_{[\mathbf{d}_{[-j,t_j]}]}, T_j, \mathbf{T}_{[-j,t_j]}, \mathbf{d}_j, \mathbf{d}_{[-j,t_j]}, t_j, \mathbf{t}_{[-j]})} \frac{p(t_j^\star, \mathbf{d}_j^\star \mid \mathbf{t}_{[-j]}, \mathbf{d}_{[-j]})p(T_j^\star)p(n_\epsilon^\star)q_b(C_j)}{p(t_j, \mathbf{d}_j \mid \mathbf{t}_{[-j]}, \mathbf{d}_{[-j]})p(T_j)p(n_\epsilon)q_b(C_j^\star)}. \tag{5.18}$$

5.1.2.4 Shift Move $(T_j) \to (T_j^\star)$

This move disturbs the pose T_j of a meta-object by adding Gaussian noise to it while the type and part associations remain unchanged. The acceptance ratio depends only on the spatial posterior predictive distributions of the objects associated to this meta-object. The proposal likelihoods cancel due to symmetry and the final ratio is

$$R_T = \frac{p(\mathbf{x}_{[\mathbf{d}_j]} \mid \mathbf{x}_{[\mathbf{d}_{[-j,t_j]}]}, T_j^\star, \mathbf{T}_{[-j,t_j]}, \mathbf{d}_{[t_j]}, \mathbf{t})p(T_j^\star)}{p(\mathbf{x}_{[\mathbf{d}_j]} \mid \mathbf{x}_{[\mathbf{d}_{[-j,t_j]}]}, T_j, \mathbf{T}_{[-j,t_j]}, \mathbf{d}_{[t_j]}, \mathbf{t})p(T_j)}. \tag{5.19}$$

5.1.2.5 Association Move (Existing Part) $(\mathbf{d}_j, \mathbf{a}) \to (\mathbf{d}_j^\star, \mathbf{a}^\star)$

This move samples the part activation and object association of an existing part k of a single meta-object j. In the IBP metaphor, this corresponds to sampling the selection of a single existing dish (part) for a single customer (meta-object instance). If the existing part is already associated with an object, we consider this object to be temporarily re-associated to the noise process (such that there are now n_ϵ noise objects). We then use Gibbs sampling to obtain one of $n_\epsilon + 1$ possible associations: either the part k is inactive ($d_{j,k} = 0$) and not associated with any object on the table, or it is active ($d_{j,k} \neq 0$) and associated with one out of n_ϵ currently available noise objects of this scene (e.g. $\mathbf{z}_i$ when $d_{j,k} = i$). The probabilities for these cases are proportional to

$$p(d_{j,k} = i) \propto \begin{cases} p(d_{j,k} = 0 \mid \mathbf{d}_{[-j,t_j,k]}, \mathbf{t})p(n_\epsilon) & i = 0 \\ p(\mathbf{z}_i \mid \mathbf{z}_{[\mathbf{d}_{[-j,t_j,k]}]}, \mathbf{T}_{[t_j]}, d_{j,k}, \mathbf{d}_{[-j,t_j,k]}, \mathbf{t}) & i \neq 0 \\ p(d_{j,k} \neq 0 \mid \mathbf{d}_{[-j,t_j,k]}, \mathbf{t})p(n_\epsilon - 1) & \end{cases} \tag{5.20}$$

5.1.2.6 Association Move (New Parts) $(\mathbf{d}_j, \mathbf{a}) \to (\mathbf{d}_j^\star, \mathbf{a}^\star)$

This move samples the associations of objects to *new* parts. For this, we use two complementary Metropolis-Hastings moves: one move increases the number of new parts by one by assigning a noise object to a new part, while the other move decreases the number of new parts by one by assigning an associated object (of a new part) to the noise process. The acceptance ratio for removing object $\mathbf{z}_i$ from a new part k of meta-object j that currently has n_+ new parts is

$$R_- = \frac{1}{p(\mathbf{z}_i \mid T_j, d_{j,k})} \frac{p(\mathbf{d}^\star_{[j,+]} \mid \mathbf{d}_{[-j,t_j]}, \mathbf{t})}{p(\mathbf{d}_{[j,+]} \mid \mathbf{d}_{[-j,t_j]}, \mathbf{t})} \frac{p(n_\epsilon + 1)}{p(n_\epsilon)} \frac{q_+}{q_-} \tag{5.21}$$

The proposal $q_- = (n_+)^{-1}$ chooses one of the new parts to be removed uniformly at random, while the reverse proposal $q_+ = (n_\epsilon + 1)^{-1}$ chooses uniformly at random one of then $n_\epsilon + 1$ noise objects to be associated to a new part. The MH move that increases the number new parts is derived similarly.

5.1.2.7 Birth Proposal $q_b(T_j, t_j, \mathbf{d}_j)$

The birth, death, and switch moves rely on a birth proposal that samples new meta-object parameters $C_j = \{T_j, t_j, \mathbf{d}_j\}$. The general idea is to sample the pose T_j and type t_j in a first step. We then proceed to sample the associations $\mathbf{d}_j$ given T_j and t_j, i.e., the potential assignment of noise objects to the parts of this meta-object.

Sampling T_j and t_j is done in either of two modes: the *object mode* or the *matching mode*. In object mode, we choose an object uniformly at random and center the meta-object pose T_j at the object's location (with random orientation) and add Gaussian noise to it. The type t_j is sampled from the current predictive distribution of the type CRP. In contrast, the matching mode was inspired by bottom-up top-down approaches and aims to propose T_j and t_j in a more efficient way by considering the currently instantiated meta-object types in the model. However, in contrast to the object mode, it cannot propose new meta-object types (new tables in the type CRP). It selects two objects and associates them to a suitable part pair. This suffices to define the pose T_j as the corresponding transformation of these parts into the scene. In detail, we first sample one of the objects $\mathbf{z}_i$ at random and choose its nearest neighbor $\mathbf{z}_j$. We match this ordered pair $\langle \mathbf{z}_i, \mathbf{z}_j \rangle$ against all ordered part pairs $\langle k_{i'}, k_{j'} \rangle$ of the meta-object types to obtain their matching probabilities p_m with respect to the parts' posterior predictive means, $\boldsymbol{\mu}_{i'}$ and $\boldsymbol{\mu}_{j'}$, of the spatial distribution, and their posterior predictive distributions, $M_{i'}$ and $M_{j'}$, over the observable object types ω_i and ω_j

$$p_m(\langle k_{i'}, k_{j'} \rangle \mid \langle \mathbf{z}_i, \mathbf{z}_j \rangle) \propto \mathcal{N}(d_\Delta \mid 0, \sigma_m^2) M_{i'}(\omega_i) M_{j'}(\omega_j). \tag{5.22}$$

Here, $d_\Delta = \|\mathbf{x}_i - \mathbf{x}_j\| - \|\boldsymbol{\mu}_{i'} - \boldsymbol{\mu}_{j'}\|$ is the residual of the objects' relative distance w.r.t. the distance of the posterior means and σ_m^2 is a fixed constant. We then sample a part pair $\langle k_{i'}, k_{j'} \rangle$ according to its matching probability to define, together with the objects $\mathbf{z}_i$ and

$\mathbf{z}_j$, the pose T_j. Finally, we add Gaussian noise to T_j. The sampled part pair implicitly defines the meta-object type t_j.

After having sampled T_j and t_j using either of these two modes, the next step is to sample the associations $\mathbf{d}_j$. For this, we randomly choose, without repetition, a noise object and use Gibbs sampling to obtain its association to either: (a) a yet unassociated part of the meta-object instance; (b) a new part of the meta-object instance; or (c) be considered a noise object. The probabilities for (a) and (b) are proportional to the spatial and object type posterior predictive distributions of the respective parts, while (c) is based on the noise IBP's base distribution.

Besides sampling from the proposal distribution we also need to be able to evaluate the likelihood $q_b(C_j)$ for sampling a given parameter set C_j. For this, we need to marginalize over the latent variables of the proposal, e.g. the binary mode variable (*object mode* or *matching mode*), the chosen object $\mathbf{z}_i$, and the chosen part pair of the *matching mode*.

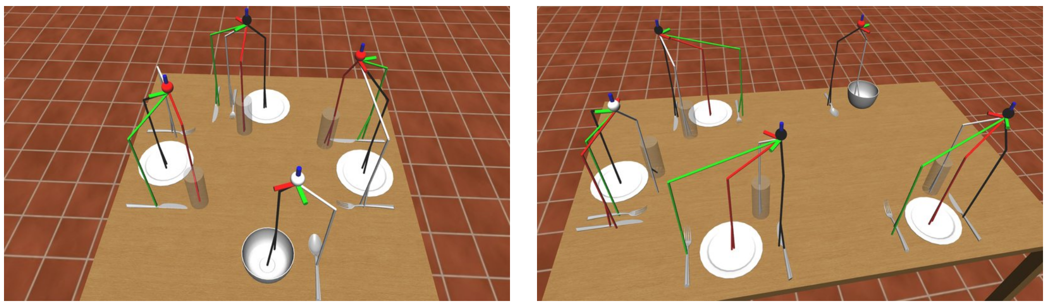
5.2 Experiments

We tested our model on both synthetic data and on real-world data acquired with a Kinect camera.

5.2.1 Synthetic Data

In the synthetic data experiment, table scenes were generated automatically using a different, hand-crafted generative model. We generated 25 training scenes ranging from two to six covers. The cover types represent two different breakfast types. The first type consists of a cereal bowl and a spoon, while the second type consists of a plate with a glass, a fork and a knife. A fork can be randomly placed at the right or the left of a plate or being absent.

The latent parameters are inferred using all the generated training scenes and setting the hyperparameters to $\alpha_\epsilon = 0.5$, $c_p = 0.25$, $\alpha_p = 2.5$, $\alpha_t = 5$, and $\alpha_c = 1$. As a first test, we wanted to show that the model is able to segment the scenes in a consistent and meaningful way. The results of this test are shown in Fig. 5.5a. A set of different scenes are segmented by using the learned model. We see that each scene has been segmented with the same cover types and objects are correctly clustered. Note that the color of the meta-objects and of the parts can change in every run, since the ordering of types and meta-objects is not relevant in our model. What is important is that the topological and metrical configuration are respected.


(a) Examples of parsed scenes of the synthetic data set.

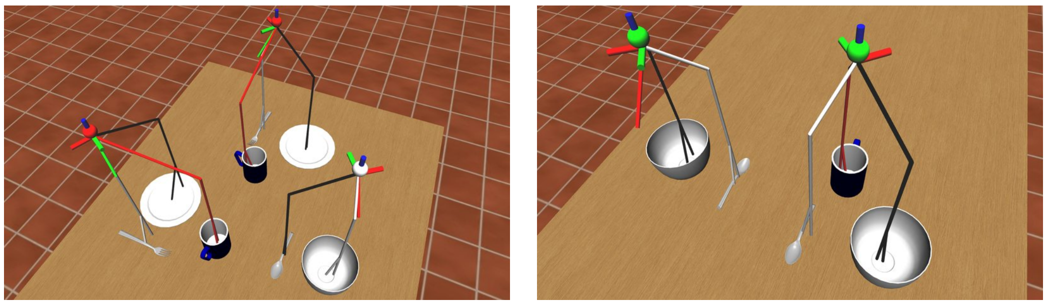
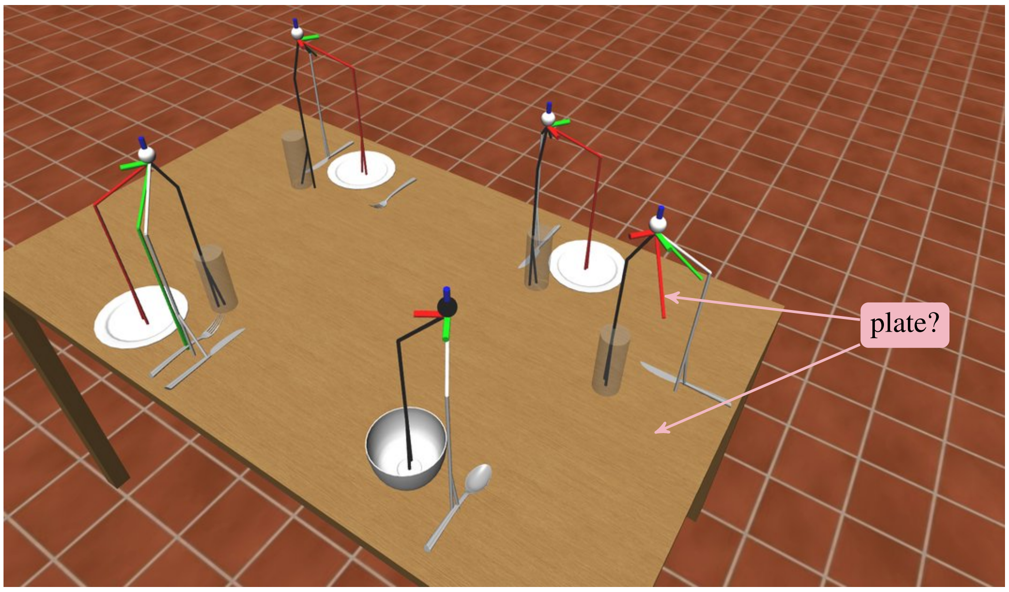

(b) Examples of parsed scenes of the real-world data set.

Figure 5.5: The picture shows the inferred meta-objects with their part relationships. Each picture shows a different scene from a different MCMC run. The color of the spheres correspond to the meta-object type. The color of the vertical lines indicates the part of the corresponding meta-object type that the object on the table is currently associated with.

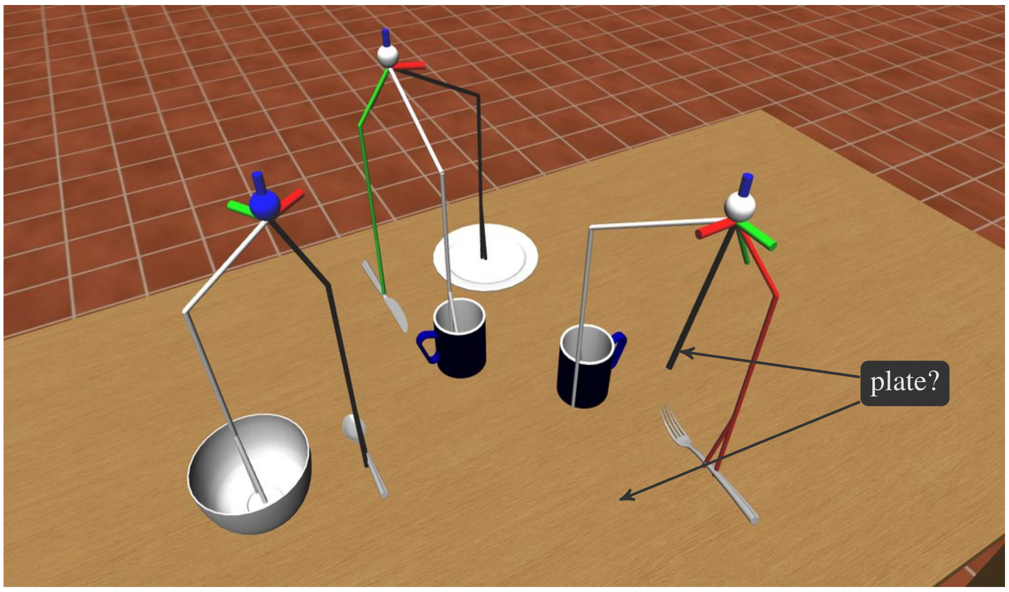
A second test is to see if the model can infer the meta-objects in incomplete scenes and if it is able to complete them. To this end, we artificially eliminated objects from already generated scenes and segmented the altered scene again using the learned model. We discovered that the approach was able to infer the correct cover type even in the absence of the missing object. As can be seen in Fig. 5.6a, the approach correctly segmented and recognized the meta-objects and is aware of the missing plate (the red branch of the hierarchy that is not grounded on an existing object). The missing object can then be inferred by sampling from the part's posterior predictive distribution over the location and type of the missing object, as the one shown in Fig. 5.7 (left) on page 124.

5.2.2 Real-world Data

We used a Microsoft Kinect depth camera to identify the objects on the table by, first, segmenting the objects by subtracting the table plane in combination with a color-based



(a) Incomplete scene of the synthetic data set.



(b) Incomplete scene of the real-world data set.

Figure 5.6: Inferred meta-objects in an incomplete scene of the synthetic data set (a) and the real-world data set (b). The color of the spheres indicate the meta-object type and the line colors indicate the part of the meta-object type. In both scenes one plate has been removed. The model is still able to correctly segment the incomplete scenes and infer the missing objects. To enhance readability the pictures have been modified with human readable labels.

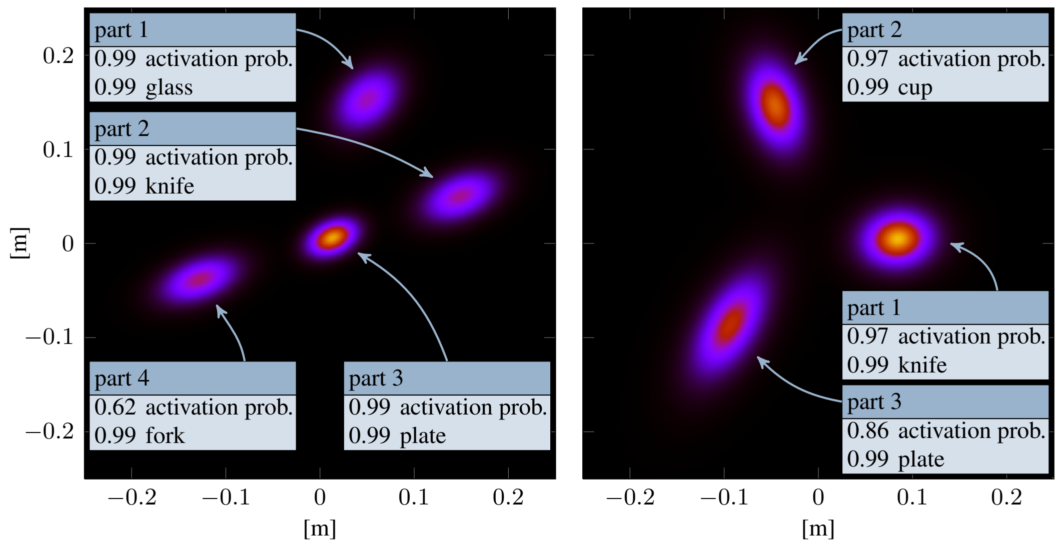


Figure 5.7: Examples of spatial posterior predictive distributions of several parts of a cover type and their activation probabilities and most likely observable object types. The cover type on the left was learned on synthetic data, while the one on the right was learned on the real-world data set.

segmentation. Second, we detect the objects based on the segmented pointclouds using a straight forward feature-based object detection system with a cascade of one-vs-all classifiers. An example for the segmentation and object identification is shown in Fig. 5.8. Note that there are likely to be better detection systems, however, the task of detecting the objects on the table is orthogonal to the scientific contribution of our proposed technique.

The same tests performed on the synthetic data have been performed also in this case, showing basically the same results. In particular, Fig. 5.5b shows the segmentation results, Fig. 5.6b shows a modified scene where a plate has been removed and Fig. 5.7 (right) shows the posterior predictive distribution for location and type, as for the synthetic case, that can be used to complete an incomplete scene.

To better illustrate the inference process, we plot the log-likelihood, the total number of parts, and the total number of meta-object types in Fig. 5.9 on page 126 as they evolve during MCMC sampling.

5.3 Related Work

In this section we describe the relevant works on unsupervised scene analysis. A first family of approaches, and the most related to our model, employs nonparametric Bayesian models

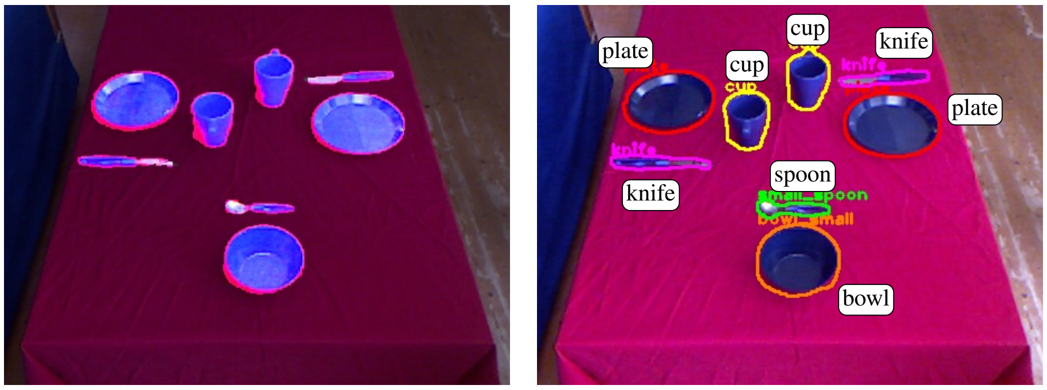


Figure 5.8: Segmentation (left) and classification (right) results on the point cloud segments visualized in the Kinect image.

to infer spatial relations. Sudderth et al. [68] introduced the transformed Dirichlet process, a hierarchical model which shares a set of stochastically transformed clusters among groups of data. The model has been applied to improve object detection in visual scenes, given the ability of the model to reason about the number of objects and their spatial configuration. Following his paper, Austerweil and Griffiths [4] presented the transformed Indian buffet process, where they model both the features representing an object and their position relative to the object center. Moreover, the set of transformations that can be applied to a feature depends on the object's context.

A complementary approach is the use of constellation models [16, 17, 53]. These models explicitly consider parts and structure of objects and can be learned efficiently and in a semisupervised manner. The star model [17], which is the more efficient variant of constellation models, uses a sparse representation of the object consisting of a star topology configuration of parts modeling the output of a variety of feature detectors. The main limitations of these methods, however, lies in the fact that the number of objects and parts must be defined beforehand and thus cannot be trivially used for scene understanding and object discovery. Ranganathan and Dellaert [53] used a 3D constellation model for representing indoor environments as object constellations. A closely related approach [50] to constellation models uses a hierarchical rule-based model to capture spatial relations. It also employs a star constellation model and a variant of the expectation maximization (EM) algorithm to infer the structure and the labels of the objects and parts.

Another family of approaches relies on discriminative learning and unsupervised model selection techniques. One approach is to automatically discover object part representa-

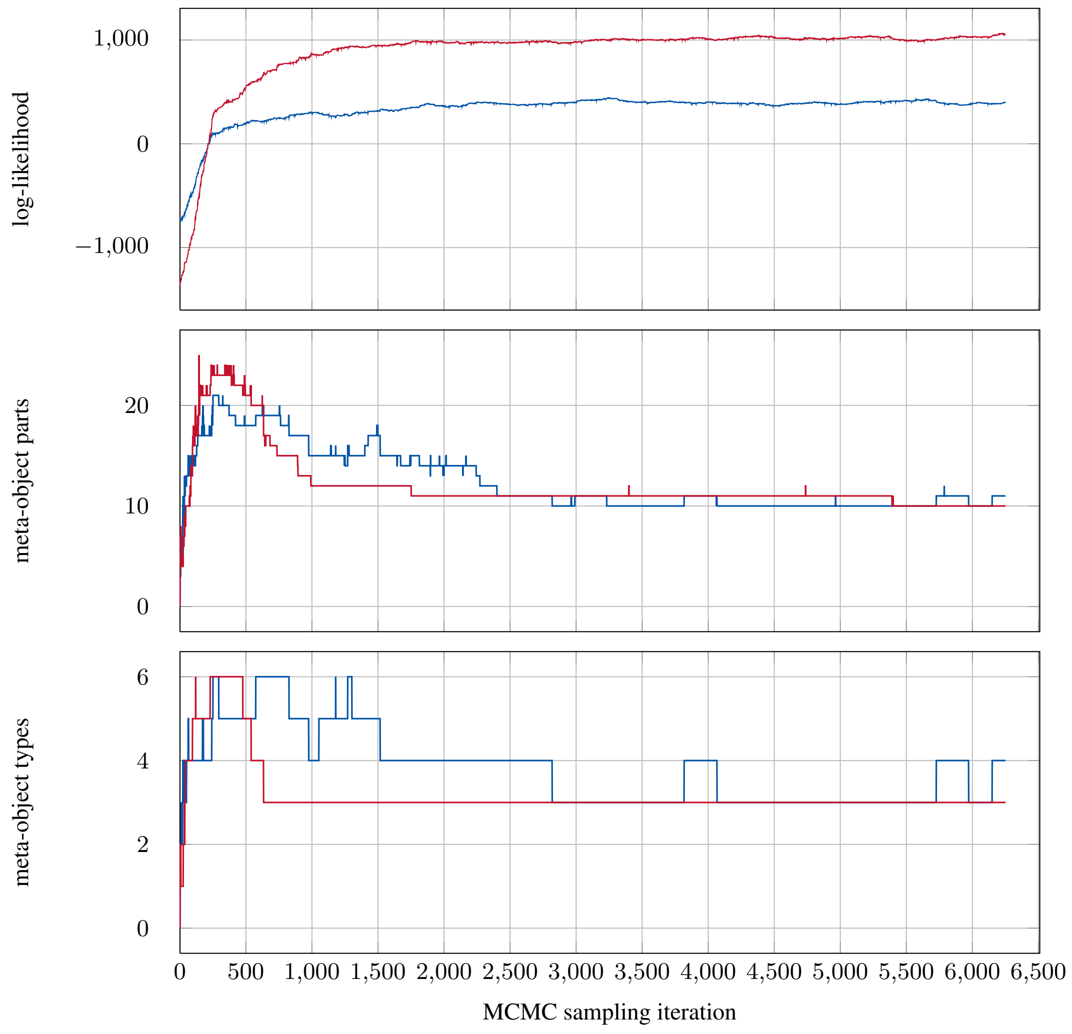


Figure 5.9: The plots depict the log-likelihood, the total number of meta-object parts (summed over all meta-object types), and the number of meta-object types as they evolve during MCMC sampling. The red lines corresponds to the synthetic data set and the blue lines to the real-world data set.

tions [61]. In this work, the authors introduced a latent conditional random field (CRF) based on a flexible assembly of parts. Individual part labels are modeled as hidden nodes and a modified version of the EM algorithm has been developed for learning the pairwise structure of the underlying graphical model. Triebel et al. [76] presented an unsupervised approach to segment 3D range scan data and to discover objects by the frequency of the appearance of their parts. The data is first segmented using graph-based clustering, then each segment is treated as a potential object part. The authors used CRFs to represent both the part graph to model the interdependence of parts with object labels, and a scene graph to smooth the object labels. Spinello et al. [66] proposed an unsupervised detection technique based on a voting scheme of image descriptors. They introduced the concept of *latticelets*: a minimal set of pairwise relations that can generalize the patterns connectivity. Conditional random fields are then used to couple low level detections with high level scene description. Jiang et al. [32] used an undirected graphical model to infer the best placement for multiple objects in a scene. Their model considers several features of object configurations, such as stability, stacking, and semantic preferences.

A different approach is the one of Fidler and Leonardis [20]. They construct a hierarchical representation of visual input using a bottom-up strategy. They learn the statistically most significant compositions of simple features with respect to more complex higher level ones. Parts are learned sequentially, layer after layer. Separate classification and grouping technique are used for the bottom and top layers to account for the numerical difference (sensor data) and semantical ones (object category).

The novelty of the approach presented in this chapter lies in the combination of a Dirichlet process and beta-Bernoulli processes which provides us with a prior for sampling or completing entire scenes.

5.4 Conclusion

In this chapter, we presented a novel and fully probabilistic generative model for unsupervised scene analysis. Our approach is able to model the spatial arrangement of objects. It maintains a nonparametric prior over scenes, which is updated by observing scenes and can directly be applied for parsing new scenes and for model completion by inferring missing objects in an incomplete scene. Our model applies a combination of a Dirichlet process and beta processes, allowing for a probabilistic treatment of the model complexity. In this way, we avoid a model selection step, which is typically intractable for the models considered here. To evaluate our approach, we successfully used our approach to infer missing objects in complex scenes.

This chapter concludes the technical description of the main contributions of this thesis. In the next chapter, we will shortly recapitulate the achieved results and discuss directions for future research.

CHAPTER 6

Discussion

We recapitulate the main contributions of this thesis: (a) RFID-based object localization, (b) searching for objects in unknown environments, and (c) learning spatial relations between objects. Further, we describe several open research questions that we have identified during our work on the techniques presented in this thesis and discuss directions for future research.

In this thesis, we presented several novel techniques for (a) localizing objects, (b) searching for objects in unknown environments, and (c) learning spatial relations between objects in an unsupervised way. Our initial motivation for addressing these topics was the assumption that if robots should be able to perform useful tasks in domestic environments in an adaptive and efficient manner, then they need to be aware of the object-related concepts that influence the structural organization of these environments. That is, robots should be able to acquire and utilize a representation of their environment that takes into account the spatial context and interdependencies between objects. We showed that such representations enable a robot to perform tasks more efficiently or to address completely new tasks. For example, a robot might reason about missing or misplaced objects or it can search more efficiently for an object if it has some prior knowledge about usual object arrangements. Further, we argued that it would be desirable that a robot is able to acquire this knowledge in an unsupervised way.

However, as a precondition for the ability to search for objects or to learn relevant object relations, a robot needs to localize the objects in the first place. As future retail products may already be equipped with RFID tags, we considered the problem of RFID-based object localization. For this, we presented a novel combined sensor model that utilizes both the received signal strength and tag detections of RFID systems for localizing the RFID tags. Further, we showed that this sensor model can also be employed in a particle filter for self-localization of the robot when given a map of tag locations. The manual acquisition of calibration data for setting up the sensor model can be a tedious and time-consuming task in practice. We therefore proposed a technique to learn such a model in an iterative and unsupervised way, which greatly simplifies this calibration phase. The learning phase iterates between two steps: (a) localizing the RFID tags with a given sensor model and (b) learning a sensor model with a given list of tag locations. As a side effect, this iterative procedure thereby not only learns a sensor model in an unsupervised way, but also estimates the tag locations. Once we have learned a sensor model we can localize RFID tags in a non-iterative way. For further evaluation of our technique, we implemented a sensor model that has been shown to be effective for WiFi localization and we compared the accuracy of both techniques. Further, we described how the WiFi model can be improved in the context of RFID localization by mapping the signal strength in pose space rather than in 2D. We performed experiments in several real-world settings and compared our approach to other state-of-the-art techniques. The results showed that our approach achieves the computational efficiency of existing sensor-centric models and an accuracy of a state-of-the-art approach that learns a location-dependent model for each tag.

Given that we can localize objects, we moved on to address the question how knowledge about usual object arrangements can be learned and represented in such a way that a robot is able to more efficiently find an object in an unknown but similarly structured environment. For this, we proposed and evaluated two novel techniques and considered the search for a product in a supermarket as an illustrative example. The first approach is a reactive search technique and emphasizes the sequential nature of the search process, which is a sequential decision making process. Being in a certain state the robot must choose among a set of available actions, i.e., at junctions in the supermarket the robot has to decide where to search next. For this, we learned a state-to-action mapping, a policy, where a state includes the currently observed objects in direction of the different aisles and the available actions correspond to the aisle the robot may choose to visit next. To learn this state-to-action mapping, we draw on ideas from imitation learning. We generated examples of optimal search behavior by computing the shortest path from a starting location to the product. In each visited state of a demonstrated example path, the robot takes a certain action and discards the other available actions in this state. Thereby, it provides positive and negative examples of state-action pairs to be taken or not, respectively. We used this data to learn a decision tree for state-action pairs, which yields a classifier-based policy representation. This resulted in compact policy representations that resemble search heuristics. These heuristics were learned on a training set of three supermarkets and were then evaluated in a fourth supermarket.

Our second approach treats the search problem as an inference problem. Thus, instead of reactively mapping observations to actions as in our first approach, we first compute a distribution over the possible locations of the target object given the thus far observed objects and the partially known structure of the environment. The robot then chooses the next goal location by trading off the probability of finding the object there with the costs of moving to this location. Each time new information becomes available, i.e., newly detected objects or newly discovered parts of the environment, the robot recomputes the distribution. Basically, this approach encodes the relevant background knowledge as expectations about how objects co-occur. However, co-occurrence of objects can only be defined with regard to a spatial context – like objects being "in the same aisle", or one object being "in the neighboring aisle" of the other. Each particular spatial context induces a different local co-occurrence model. Motivated by the idea of combining an ensemble of base classifiers to form a more robust classifier, we used a diverse set of local co-occurrence models, each considering a different spatial relation, and fused their outcomes as features in a maximum entropy model, which in our case models the discrete distribution over all possible locations of the target object.

We performed extensive experiments and compared both of our search techniques to a baseline approach and to the performance of humans. Our evaluation showed, that the inference-based search technique yielded shorter search paths than the approach based on search heuristics. Both search techniques significantly outperform a baseline strategy that explores the environment until it finds the product but does not consider any other information except the distinction between visited and non-visited areas. This demonstrates the benefits of utilizing background knowledge when searching for objects in unknown environments. We additionally compared our approaches to the search efficiency of human subjects in a real supermarket. Roughly speaking, our inference-based search technique was able to halve the average search path length when compared to the exploration strategy, while the humans were able to top our approaches by again halving the average search path length when compared to our inference-based search technique.

Our inference-based search technique basically relied on object co-occurrence statistics. However, the spatial context within which objects could co-occur was manually given by the predefined spatial relations – like objects being "in the same aisle" or "in the same shelf”. In our last contribution of this thesis, we took the learning scenario one step further by simultaneously learning both the spatial context and the co-occurrence of objects within a spatial context. That is, we learned spatially coherent object constellations in multi-object scenes in an unsupervised way. As an application scenario we considered tabletop scenes and our goal was to learn the different object constellations that correspond to the place covers consisting of, e.g., a plate, a knife, and a mug. For this, we presented a novel and fully probabilistic generative model for unsupervised scene analysis. Our approach maintains a nonparametric prior over scenes, which is updated by observing scenes and can directly be applied for parsing new scenes and for model completion by inferring missing objects in an incomplete scene. Our model is based on nonparametric Bayesian models in form of the Dirichlet process and the beta-Bernoulli process. This allows for a probabilistic treatment of the model complexity and we thereby avoided a model selection step, which is typically intractable for the models considered here. Further, this also has practical benefits in the context of lifelong learning for service robots. As the number of constellation types is not fixed in our model, the robot is able to recognize and integrate previously unseen constellations into its model in an open-ended fashion and within a single coherent probabilistic framework. Finally, we evaluated our approach and successfully used our approach to infer missing objects in complex scenes.

6.1 Outlook

During our work on the techniques presented in this thesis, we identified several open research question and directions for future work that we did not address within the scope of this thesis and that we would like to discuss in the following.

6.1.1 RFID-based Object Localization and Self-Localization

The results of our RFID-based localization technique showed that the combined consideration of signal strength information along with tag detection events lead to an improved sensor-centric model. On the other hand, signal strength maps based on Gaussian process regression resulted in a comparable accuracy as our combined sensor-centric model. Therefore, it would be interesting to see if the accuracy of signal strength maps could also be improved in just the same way by combining them with tag detection maps that would model the probability of detecting a tag at a specific location.

We only considered 2D localization of the RFID tags and we placed the RFID tags at the same height above the ground. It is straightforward to extend our technique to the case of 3D localization by learning a 3D sensor model and estimating the position of the tags in 3D. It would be interesting to see if this leads to new challenges and how this would affect the accuracy.

Moreover, the assumption that the tags remain static could be alleviated. For example, we could also assume that some tags are semi-static and change their location every now an then, such as a coffee cup in an office building. This would pose new challenges to our estimation technique and, on a technical level, resembles the kidnapped robot problem in which a robot is suddenly relocated and has to adapt its belief over its position accordingly.

Further, it would be interesting to incorporate priors derived from other sensor modalities when localizing RFID tags. For example, we could use a grid map estimated from laser range data to incorporate a prior that models the assumption that tags are attached to walls, similar as done in [58]. Or we could use visual information if we have prior information about how the tags look like. However, the reliance on visual information would impair one of the benefits of using RFID tags, namely the fact that they can be detected without being in the line-of-sight.

6.1.2 Searching for Objects

The reactive search strategy learned the decision trees based on isolated examples of aisles that have been taken or not at junctions during demonstrated optimal search paths. Considering isolated examples of positive and negative edges facilitated data generation and learning the decision trees was straightforward. The learned search heuristics achieved significantly shorter search paths. However, our actual goal is not to minimize the error rate of a classifier that classifies aisles into promising or non-promising directions, but to minimize the overall search path length. Hence, during the learning phase, it would be interesting to directly search in the space of heuristics, i.e., the space of decision trees. A given search heuristic could be evaluated by simulating several search runs in the training markets. This could lead to improved search heuristics as they are directly optimized in terms of the resulting path lengths rather than the error rate of a classifier. Further, the application of a learned search heuristic during the actual search would remain unchanged and thus would be equally efficient.

For the inference-based strategy, we fused the outcomes of several basic models in a maximum entropy model. It would be interesting to evaluate another approach that directly uses the features of the base models in the maxent model. Further, we currently select a target location and then plan the shortest path to this location. While we argued that planning optimal search paths is in general computationally infeasible, we might at least consider to sample a few paths to a target location and trade-off the distance and occurrence probabilities *along* the sampled paths, instead of just considering the occurrence probability at the final location.

6.1.3 Learning Object Constellations

There are also several ways to extend our work on learning object constellations. First of all, we only modeled 2D constellations due to our motivation to learn objects arrangements on a table which inherently is a 2D problem. However, our model is quite general and is not specifically geared towards 2D data. In our current implementation, the spatial constraints are modeled as 2D Gaussian distributions. The inclusion of an additional third dimension would therefore pose no relevant computational overhead. However, it would be necessary to adapt the birth proposal, which indeed would affect the computational efficiency. The birth proposal proposed a transformation of the meta-object's reference frame into the scene by randomly selecting two objects on the table and associating them to two parts of a meta-object. For the 3D case, we would need to select three reference objects to efficiently propose a transformation into the scene. This impacts the computational efficiency, because the birth proposal relies on computing a matching probability between the selected objects and the parts of meta-objects. By increasing the number of the to-be-matched objects by one we thereby increase the complexity for computing the matching probabilities from quadratic to cubic with respect to the number of *parts* of a meta-object. However, the complexity will stay linear with respect to the number of meta-object *types*, as we require that the matched parts all stem from the same meta-object type and we therefore only have to evaluate all part-triplets "within" each meta-object type.

Next, we could extend the learning scenario to multiple levels and learn one layer at a time. That is, the learned constellations could be the input for the next level which then learns meta-meta-objects as constellations of constellations. However, this again would require a slight change in the model, because we assumed that the objects are labeled 2D *points*, but the learned meta-objects are labeled 2D *poses*. If we aim at extending the work towards a hierarchical layer-wise learning framework, we therefore are in need of a spatial distribution defined over the pose space. A simple approach would be to assume independence between the location and orientation of an object, and to place a von Mises prior on the orientation.

Further, instead of extending the model to a higher level that learns constellations of objects constellations, we could also try the opposite direction and try to detect objects as constellations of object parts. It particular, it would be interesting to start from basic sensory information such as pixels or point clouds. In this case, we would have to rework our prior over the input vocabulary. In our case, this was a Dirichlet distribution over the observable object types, such as plates and knifes. For dealing with basic sensory input, we would have to replace this prior with an appropriate distribution over pixel patches or small subsets of colored point clouds. Another idea would be to extend our approach to a spatio-temporal model for unsupervised activity recognition, such as waving one's hand, walking, jumping, etc.

Next, we could add more global moves to our sampler to avoid that the Markov chain gets stuck in local modes of the posterior distribution. In our current implementation, the MCMC moves operate on the level of individual meta-object *instances* by adding, changing, or removing instances. While this means that we are already changing the associations of several objects at once, one could image even more global moves that operate at the level of meta-object *types*. We could thereby influence all of the corresponding meta-object instances across all scenes. One idea would be to simultaneously change the pose of all meta-object instances of the same type. In effect, this would correspond to translating and rotating the reference frame of a meta-object type with respect to its parts. Roughly speaking, if the reference frames of meta-object type is shifted towards the center of mass of its parts, then this would increase the data likelihood. That is because the normal-Wishart prior gives higher likelihood to parts close to the origin of the reference frame and a lower likelihood to the ones further away. If the reference frame is closer to the center of mass of its parts then this would also decrease the likelihood of wrongly associating objects of other nearby object constellations during subsequent MCMC moves. Another idea for introducing more global moves would be to add split-and-merge moves that either merge all instances of two types into one type, or, for the reverse case, split the instances of a certain type into two types. In the CRP metaphor this would correspond to redistributing the customers (meta-object instances) of one table of the type CRP on two tables (split move), or to joining the customers of two tables at one table (merge move). Methods for general conjugate [28] and nonconjugate [29] DP mixture models have already been published. However, without going into details, we like to stress that it is quite likely that split-and-merge moves entail a high computational burden for our model, because each affected meta-object instance during such a move requires a re-assignment of its objects to the parts of its newly assigned meta-object type.

Next, our current model does not consider dependencies between the poses of meta-object instances across scenes. However, if a robot would see two scenes of the very same place, say from two different days, then it certainly makes sense to assume a dependency between these related scenes. For example, if at day one a cover is placed at a certain location on the breakfast table, e.g., in front of a chair, then it is quite likely that at day two the robot will again observe a cover at roughly the same place and with probably the same type. Hence, the model of a scene should include "places" at which meta-object instances might occur. Formally, a place should be modeled as a distribution over the poses and types of the meta-object instances that we expect to see there and, further, it should include a binary probability for the existence or non-existence of an instance at this place in a given scene. Luckily, this can easily be incorporated into our model. Remember, that in our current model the type and pose of the meta-objects for a scene are sampled from a scene-specific Indian buffet process and there is just a single customer in this IBP: the scene itself. This basically means that we are drawing the number of meta-objects of this scene from a Poisson distribution, the individual types from the global type CRP, and the poses from a uniform distribution that depends on the table area. To introduce the desired dependencies, scenes would become customers in a IBP that is shared among *related* scenes. The dish parameters of this IBP would need to be defined differently and would then correspond to the parameters of a place instead of the parameters of a meta-object instance. That is, a dish parameter would no longer directly correspond to the type and pose of a single meta-object instance in one scene but rather to a distribution over the poses and types of meta-object instances at this place in any of the related scenes. Further, we need a prior from which these distributions are sampled. The prior for a pose distribution should again depend on the table surface, but we will not go into details here. The prior for a type distributions could be a Dirichlet process where the base distribution is the global distribution over types. As usual, the draws from this process can be integrated out which results in each place having its own type CRP. This corresponds to a hierarchical CRP in which a table of a type CRP of a *place* references a table in the *global* type CRP.

6.2 Concluding Remarks

In this thesis, we presented several techniques that enable service robots to represent, learn, and utilize the immanent object-related concepts of domestic environments. We mainly focused on three topics: (a) RFID-based object localization and self-localization, (b) modeling and utilizing background knowledge about usual object arrangements to more efficiently find objects in unknown environments, and (c) unsupervised learning of object arrangements from everyday scenes.

These techniques are preliminary steps towards the long-term goal of building autonomous service robots. We believe, that especially nonparametric Bayesian models can be a useful tool towards this long-term goal. The rationale for this is, that these models enable an autonomous service robot to deal with lifelong learning scenarios in which a robot has to classify each new observation as being an instance of either a known concept or of a completely new concept, which in the latter case could then be integrated into the model. In principle, this enables a robot to learn new concepts in an open-ended fashion and within a single coherent probabilistic framework.

Appendix

A.1 Posterior Predictive Distribtion w.r.t. a Normal-Wishart Prior

The following description is based on the formulas given in [45]. We have a data set $\{\mathbf{x}_{1:n}\}$ of n points with $\mathbf{x}_i \in \mathbb{R}^d$. We assume that these points have been sampled from a Gaussian distribution which in turn has been sampled from a normal-Wishart distribution. The Gaussian is parametrized by an unknown mean and an unknown precision matrix. We analytically integrate out the parameters of the Gaussian to yield a posterior predictive distribution $p(\mathbf{x}^* \mid \mathbf{x}_{1:n})$ over a new data point $\mathbf{x}^*$ given the previously seen points and the parameters of the normal-Wishart prior, which are $\boldsymbol{\mu}_0 \in \mathbb{R}^d$, $\nu_0 \in \mathbb{R}$, $\kappa_0 \in \mathbb{R}$, and $\mathbf{T}_0 \in \mathbb{R}^{d\times d}$, where the latter is a prior covariance matrix. We denote the mean of the data points by

$$\overline{\mathbf{x}} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i \tag{A.1}$$

and the scatter matrix of the data points by

$$\mathbf{S} = \sum_{i=1}^{n}(\mathbf{x}_i - \overline{\mathbf{x}})(\mathbf{x}_i - \overline{\mathbf{x}})^\mathsf{T}. \tag{A.2}$$

Then the updated parameters of the normal-Wishart prior based on the data set are

$$\boldsymbol{\mu}_n = \frac{\kappa_0 \boldsymbol{\mu}_0 + n\overline{\mathbf{x}}}{\kappa_0 + n} \tag{A.3}$$

$$\mathbf{T}_n = \mathbf{T}_0 + \mathbf{S} + \frac{\kappa_0 n}{\kappa_0 + n}(\boldsymbol{\mu}_0 - \overline{\mathbf{x}})(\boldsymbol{\mu}_0 - \overline{\mathbf{x}})^\mathsf{T} \tag{A.4}$$

$$\kappa_n = \kappa_0 + n \tag{A.5}$$

$$\nu_n = \nu_0 + n. \tag{A.6}$$

Then the posterior predictive distribution, that we are interested in, is

$$p(\mathbf{x}^* \mid \mathbf{x}_{1:n}) = t_{\nu_n - d + 1}\left(\boldsymbol{\mu}_n, \frac{\mathbf{T}_n(\kappa_n + 1)}{\kappa_n(\nu_n - d + 1)}\right). \tag{A.7}$$

Here, $t_m(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ denotes a d-dimensional multivariate t-distribution with m degrees of freedom, location parameter $\boldsymbol{\mu} \in \mathbb{R}^d$, and scale matrix $\boldsymbol{\Sigma} \in \mathbb{R}^{d\times d}$. This distribution has the following analytical form

$$t_m(\mathbf{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{\Gamma(\frac{m}{2} + \frac{d}{2})}{\Gamma(\frac{m}{2})} \frac{|\boldsymbol{\Sigma}|^{-\frac{1}{2}}}{(m\pi)^{\frac{d}{2}}} \left(1 + \frac{1}{m}(\mathbf{x} - \boldsymbol{\mu})^\mathsf{T} \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu})\right)^{-\frac{m+d}{2}}, \tag{A.8}$$

where $\Gamma(\cdot)$ is the gamma function.

A.2 Derivations for the MaxEnt Model (Part 1)

We need to find the parameters of $h(\mathbf{q}, \boldsymbol{\lambda})$ by setting its gradient to zero. Setting the partial derivation for the Lagrange multiplier λ_0 of the normalization constraint to zero recovers the original constraint

$$\frac{\partial}{\partial \lambda_0} h(\mathbf{q}, \boldsymbol{\lambda}) = 0 \tag{A.9}$$

$$\Leftrightarrow \quad -\left(\sum_i q_i - 1\right) = 0 \tag{A.10}$$

$$\Leftrightarrow \quad \sum_i q_i = 1. \tag{A.11}$$

Likewise, setting the partial derivation for a feature weight λ_j to zero recovers the constraint that the expected value of this feature function should match the value E_j

$$\frac{\partial}{\partial \lambda_j} h(\mathbf{q}, \boldsymbol{\lambda}) = 0 \qquad \text{for each } j = 1, \ldots, m \tag{A.12}$$

$$\Leftrightarrow \qquad -\left(E_j - \sum_i q_i f_j(x_i)\right) = 0 \tag{A.13}$$

$$\Leftrightarrow \qquad \sum_i q_i f_j(x_i) = E_j. \tag{A.14}$$

The partial derivation for each probability parameter q_i is

$$\frac{\partial}{\partial q_i} h(\mathbf{q}, \boldsymbol{\lambda}) = -\left(\frac{\partial}{\partial q_i}\left(q_i \log q_i\right)\right) - \left(\frac{\partial}{\partial q_i}(\lambda_0 q_i)\right) + \left(\frac{\partial}{\partial q_i}\left(\sum_j \sum_{i'} \lambda_j q_{i'} f_j(x_{i'})\right)\right) \tag{A.15}$$

$$= -\log q_i - q_i \frac{1}{q_i} - \lambda_0 + \sum_j \lambda_j f_j(x_i) \tag{A.16}$$

$$= -\log q_i - 1 - \lambda_0 + \sum_j \lambda_j f_j(x_i). \tag{A.17}$$

Setting this to zero leads to

$$-\log q_i - 1 - \lambda_0 + \sum_j \lambda_j f_j(x_i) = 0 \tag{A.18}$$

$$\Leftrightarrow \qquad \log q_i = -1 - \lambda_0 + \sum_j \lambda_j f_j(x_i) \tag{A.19}$$

$$\Leftrightarrow \qquad q_i = \exp\left(-1 - \lambda_0 + \sum_j \lambda_j f_j(x_i)\right). \tag{A.20}$$

We can use the result of Eq. (A.20) to expand the normalization constraint of Eq. (A.11) which leads to

$$\sum_i q_i = 1 \tag{A.21}$$

$$\Leftrightarrow \qquad \sum_i \exp\left(-1-\lambda_0+\sum_j \lambda_j f_j(x_i)\right) = 1 \tag{A.22}$$

$$\Leftrightarrow \qquad \exp(-1-\lambda_0)\sum_i \exp\left(\sum_j \lambda_j f_j(x_i)\right) = 1 \tag{A.23}$$

$$\Leftrightarrow \qquad \exp(-1-\lambda_0) = \frac{1}{\sum_i \exp\left(\sum_j \lambda_j f_j(x_i)\right)} \tag{A.24}$$

Now we can use the result of Eq. (A.24) to expand Eq. (A.20) which leads to

$$q_i = \exp\left(-1-\lambda_0+\sum_j \lambda_j f_j(x_i)\right) \tag{A.25}$$

$$= \underbrace{\exp\left(-1-\lambda_0\right)}_{\text{see Eq. (A.24)}} \exp\left(\sum_j \lambda_j f_j(x_i)\right) \tag{A.26}$$

$$= \frac{\exp\left(\sum_j \lambda_j f_j(x_i)\right)}{\sum_{i'} \exp\left(\sum_j \lambda_j f_j(x_{i'})\right)}. \tag{A.27}$$

With Eq. (A.27) we now have the final analytic form for the discrete probability distribution.

A.3 Derivations for the MaxEnt Model (Part 2)

The gradient of $\ell(\boldsymbol{\lambda}_{1:m})$ is

$$\nabla \ell(\boldsymbol{\lambda}_{1:m}) = \nabla \frac{1}{n_s}\sum_s \left(\log \psi(x^{(s)}) - \log \sum_x \psi(x)\right) \tag{A.28}$$

$$= \frac{1}{n_s}\sum_s \left(\nabla \log \psi(x^{(s)}) - \frac{\nabla \sum_x \psi(x)}{\sum_{x'} \psi(x')}\right) \tag{A.29}$$

$$= \frac{1}{n_s}\sum_s \left(\mathbf{f}(x^{(s)}) - \frac{\sum_x \psi(x)\mathbf{f}(x)}{\sum_{x'} \psi(x')}\right) \tag{A.30}$$

$$= \frac{1}{n_s}\sum_s \left(\mathbf{f}(x^{(s)}) - \sum_x \underbrace{\frac{\psi(x)}{\sum_{x'} \psi(x')}}_{p(x)} \mathbf{f}(x)\right) \tag{A.31}$$

$$= \frac{1}{n_s} \sum_s \left(\mathbf{f}(x^{(s)}) - \sum_x p(x) \mathbf{f}(x) \right) \tag{A.32}$$

$$= \frac{1}{n_s} \sum_s \Big(\mathbf{f}(x^{(s)}) - \mathbb{E}_{p(x)} \left[\mathbf{f}(x) \right] \Big) \tag{A.33}$$

$$= \frac{1}{n_s} \left(\left(\sum_s \mathbf{f}(x^{(s)}) \right) - n_s \mathbb{E}_{p(x)} \left[\mathbf{f}(x) \right] \right) \tag{A.34}$$

$$= \mathbb{E}_{\tilde{p}(x)}[\mathbf{f}(x)] - \mathbb{E}_{p(x)} \left[\mathbf{f}(x) \right]. \tag{A.35}$$

List of Figures

List of Tables

Bibliography

[1] Cesare Alippi, Dario Cogliati, and Giovanni Vanini. A statistical approach to localize passive RFIDs. In *Proc. of the IEEE Int. Symp. on Circuits and Systems (ISCAS)*, pages 843–846, Island of Kos, Greece, 2006. ISBN 0-7803-9389-9. doi:10.1109/ISCAS.2006.1692717.

[2] Christophe Andrieu, Nando de Freitas, Arnaud Doucet, and Michael I. Jordan. An introduction to MCMC for machine learning. *Machine Learning*, 50(1–2):5–43, 2003. doi:10.1023/A:1020281327116.

[3] Brenna D. Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 57(5):469–483, 2009. doi:10.1016/j.robot.2008.10.024.

[4] Joseph L. Austerweil and Thomas L. Griffiths. Learning invariant features using the transformed Indian buffet process. In *Advances in Neural Information Processing Systems (NIPS)*, pages 82–90, 2010. URL http://books.nips.cc/papers/files/nips23/NIPS2010_0437.pdf.

[5] Stanley J. Benkoski, Michael G. Monticino, and James R. Weisinger. A survey of the search theory literature. *Naval Research Logistics (NRL)*, 38(4):469–494, 1991. doi:10.1002/1520-6750(199108)38:4<469::AID-NAV3220380404>3.0.CO;2-E.

[6] Adam L. Berger, Vincent J. Della Pietra, and Stephen A. Della Pietra. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71, 1996.

[7] David M. Blei, Thomas L. Griffiths, and Michael I. Jordan. The nested Chinese restaurant process and Bayesian nonparametric inference of topic hierarchies. *Journal of the ACM*, 57(2), 2010. doi:10.1145/1667053.1667056.

[8] Leonard A. Breslow and David W. Aha. Simplifying decision trees: A survey. *The Knowledge Engineering Review*, 12(1):1–40, 1997.

[9] Thimothy H. Chung and Joel W. Burdick. A decision-making framework for control strategies in probabilistic search. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 4386–4393, Roma, Italy, 2007. doi:10.1109/ROBOT.2007.364155.

[10] Alexandru Cocora, Kristian Kersting, Christian Plagemann, Wolfram Burgard, and Luc De Raedt. Learning relational navigation policies. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 2792–2797, Beijing, China, 2006. doi:10.1109/IROS.2006.282061.

[11] Frank Dellaert, Dieter Fox, Wolfram Burgard, and Sebastian Thrun. Monte Carlo localization for mobile robots. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, volume 2, pages 1322–1328, 1999. ISBN 0-7803-5180-0. doi:10.1109/ROBOT.1999.772544.

[12] Travis Deyle, Charles C. Kemp, and Matthew S. Reynolds. Probabilistic UHF RFID tag pose estimation with multiple antennas and a multipath RF propagation model. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 1379–1384, Nice, France, 2008. doi:10.1109/IROS.2008.4651170.

[13] Travis Deyle, Hai Nguyen, Matt Reynolds, and Charles C. Kemp. RF vision: RFID receive signal strength indicator (RSSI) images for sensor fusion and mobile manipulation. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 5553–5560, St. Louis, MO, USA, 2009. doi:10.1109/IROS.2009.5354047.

[14] Arnaud Doucet, Nando de Freitas, and Neil Gordon, editors. *Sequential Monte Carlo Methods in Practice*. Springer, 2001.

[15] Isaac Ehrenberg, Christian Floerkemeier, and Sanjay Sarma. Inventory management with an RFID-equipped mobile robot. In *Proc. of the IEEE Int. Conf. on Automation Science and Engineering (CASE)*, pages 1020–1026, 2007. ISBN 978-1-4244-1154-2. doi:10.1109/COASE.2007.4341838.

[16] R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages II–264–II–271, 2003. doi:10.1109/CVPR.2003.1211479.

[17] R. Fergus, P. Perona, and A. Zisserman. A sparse object category model for efficient learning and exhaustive recognition. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 380–387, 2005. doi:10.1109/CVPR.2005.47.

[18] Thomas S. Ferguson. A Bayesian analysis of some nonparametric problems. *The Annals of Statistics*, 1(2):209–230, 1973. doi:10.1214/aos/1176342360.

[19] Brian Ferris, Dirk Hähnel, and Dieter Fox. Gaussian processes for signal strength-based location estimation. In *Proc. of Robotics: Science and Systems (RSS)*, Philadelphia, PA, USA, 2006.

[20] Sanja Fidler and Aleš Leonardis. Towards scalable representations of object categories: Learning a hierarchy of parts. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, 2007. doi:10.1109/CVPR.2007.383269.

[21] Dieter Fox, Sebastian Thrun, Wolfram Burgard, and Frank Dellaert. Particle filters for mobile robot localization. In Arnaud Doucet, Nando de Freitas, and Neil Gordon, editors, *Sequential Monte Carlo Methods in Practice*, chapter 19. Springer, 2001.

[22] Cipriano Galindo, Juan-Antonio Fernández-Madrigal, Javier González, and Alessandro Saffiotti. Robot task planning using semantic maps. *Robotics and Autonomous Systems*, 56(11):955–966, 2008. doi:10.1016/j.robot.2008.08.007.

[23] Samuel J. Gershman and David M. Blei. A tutorial on Bayesian non-parametric models. *Journal of Mathematical Psychology*, 56(1):1–12, 2012. doi:10.1016/j.jmp.2011.08.004.

[24] Thomas L. Griffiths and Zoubin Ghahramani. Infinite latent feature models and the Indian buffet process. In *Advances in Neural Information Processing Systems (NIPS)*, pages 475–482. 2006. URL http://books.nips.cc/papers/files/nips18/NIPS2005_0130.pdf.

[25] Giorgio Grisetti, Cyrill Stachniss, and Wolfram Burgard. Improved techniques for grid mapping with Rao-Blackwellized particle filters. *IEEE Trans. on Robotics*, 23 (1):34–46, 2007. doi:10.1109/TRO.2006.889486.

[26] Dirk Hähnel, Wolfram Burgard, Dieter Fox, Ken Fishkin, and Matthai Philipose. Mapping and localization with RFID technology. In *Proc. of the IEEE Int. Conf.*

on Robotics and Automation (ICRA), volume 1, pages 1015–1020, New Orleans, LA, USA, 2004. doi:10.1109/ROBOT.2004.1307283.

[27] Tom Heskes. Selecting weighting factors in logarithmic opinion pools. In *Advances in Neural Information Processing Systems (NIPS)*, pages 266–272, 1998.

[28] Sonia Jain and Radford M. Neal. A split-merge Markov chain Monte Carlo procedure for the Dirichlet process mixture model. *Journal of Computational and Graphical Statistics*, 13(1):158–182, 2004. doi:10.1198/1061860043001.

[29] Sonia Jain and Radford M. Neal. Splitting and merging components of a nonconjugate Dirichlet process mixture model. *Bayesian Analysis*, 2(3):445–472, 2007. doi:10.1214/07-BA219.

[30] Edwin T. Jaynes. Information theory and statistical mechanics. *The Physical Review*, 106(4):620–630, 1957.

[31] Edwin T. Jaynes. *Probability Theory: The Logic of Science*. Cambridge University Press, 2003.

[32] Yun Jiang, Marcus Lim, Changxi Zheng, and Ashutosh Saxena. Learning to place new objects in a scene. *Int. Journal of Robotics Research (IJRR)*, 31(9):1021–1043, 2012. doi:10.1177/0278364912438781.

[33] Dominik Joho, Christian Plagemann, and Wolfram Burgard. Modeling RFID signal strength and tag detection for localization and mapping. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 3160–3165, Kobe, Japan, May 2009. ISBN 978-1-4244-2788-8. doi:10.1109/ROBOT.2009.5152372.

[34] Michael I. Jordan. Hierarchical models, nested models and completely random measures. In Ming-Hui Chen, Dipak K. Dey, Peter Müller, Dongchu Sun, and Keying Ye, editors, *Frontiers of Statistical Decision Making and Bayesian Analysis: In Honor of James O. Berger*, chapter 6.3, pages 207–218. Springer, 2010.

[35] Christopher Kalff and Gerhard Strube. Where is the fresh yeast? The use of background knowledge in human navigation. In *Spatial Cognition 2008: Poster Presentations*, pages 17–20, Freiburg, Germany, 2008.

[36] Takayuki Kanda, Masahiro Shiomi, Laurent Perrin, Tatsuya Nomura, Hiroshi Ishiguro, and Norihiro Hagita. Analysis of people trajectories with ubiquitous sensors in a science museum. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 4846–4853, Roma, Italy, 2007. ISBN 1-4244-0601-3. doi:10.1109/ROBOT.2007.364226.

[37] Alexander Kleiner, Christian Dornhege, and Sun Dali. Mapping disaster areas jointly: RFID-coordinated SLAM by humans and robots. In *Proc. of the IEEE Int. Workshop on Safety, Security and Rescue Robotics (SSRR)*, Roma, Italy, 2007.

[38] Sven Koenig. Agent-centered search. *AI Magazine*, 22(4):109–131, 2001. ISSN 0738-4602.

[39] Thomas Kollar and Nicholas Roy. Utilizing object-object and object-scene context when planning to find things. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 2168–2173, Kobe, Japan, 2009. doi:10.1109/ROBOT.2009.5152831.

[40] Niels Landwehr, Bernd Gutmann, Ingo Thon, Matthai Philipose, and Luc De Raedt. Relational transformation-based tagging for human activity recognition. *Fundamenta Informaticae*, 89(1):111–129, 2008. ISSN 0169-2968.

[41] Haye Lau, Shoudong Huang, and Gamini Dissanayake. Optimal search for multiple targets in a built environment. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 3740–3745, Edmonton, AB, Canada, 2005. doi:10.1109/IROS.2005.1544986.

[42] Jane Liu and Mike West. Combined parameter and state estimation in simulation-based filtering. In Arnaud Doucet, Nando de Freitas, and Neil Gordon, editors, *Sequential Monte Carlo Methods in Practice*, chapter 10. Springer, 2001.

[43] Xiaotao Liu, Mark Corner, and Prashant Shenoy. Ferret: RFID localization for pervasive multimedia. In *Proc. of the Int. Conf. on Ubiquitous Computing (UbiComp)*, 2006.

[44] Alexei A. Makarenko, Stefan B. Williams, Frederic Bourgault, and Hugh F. Durrant-Whyte. An experiment in integrated exploration. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 534–539, Lausanne, Switzerland, 2002. doi:10.1109/IRDS.2002.1041445.

[45] Kevin P. Murphy. Conjugate Bayesian analysis of the Gaussian distribution. Self-published notes, 2007. URL http://www.cs.ubc.ca/~murphyk/Papers/bayesGauss.pdf.

[46] Radford M. Neal. Probabilistic inference using Markov chain Monte Carlo methods. Technical Report CRG-TR-93-1, Department of Computer Science, University of Toronto, 1993.

[47] Radford M. Neal. Markov chain sampling methods for Dirichlet process mixture models. *Journal of Computational and Graphical Statistics*, 9(2):249–265, 2000. doi:10.1080/10618600.2000.10474879.

[48] Lionel M. Ni, Yunhao Liu, Yiu Cho Lau, and Abhishek P. Patil. LANDMARC: Indoor location sensing using active RFID. In *Proc. of the IEEE Int. Conf. on Pervasive Computing and Communications (PerCom)*, pages 407–415, 2003. doi:10.1109/PERCOM.2003.1192765.

[49] John Paisley, David M. Blei, and Michael I. Jordan. Stick-breaking beta processes and the Poisson process. In *Proc. of the Int. Conf. on Artificial Intelligence and Statistics (AISTATS)*, 2012. URL http://jmlr.org/proceedings/papers/v22/paisley12/paisley12.pdf.

[50] Devi Parikh, C. Lawrence Zitnick, and Tsuhan Chen. Unsupervised learning of hierarchical spatial structures in images. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 2743–2750, 2009. doi:10.1109/CVPR.2009.5206549.

[51] M. Philipose, K.P. Fishkin, M. Perkowitz, D.J. Patterson, D. Fox, H. Kautz, and D. Hähnel. Inferring activities from interactions with objects. *IEEE Pervasive Computing*, 3(4):50–57, October 2004. doi:10.1109/MPRV.2004.7.

[52] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986. doi:10.1007/BF00116251.

[53] Ananth Ranganathan and Frank Dellaert. Semantic modeling of places using objects. In *Proc. of Robotics: Science and Systems (RSS)*, Atlanta, GA, USA, 2007. URL http://www.roboticsproceedings.org/rss03/p01.pdf.

[54] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006. ISBN 0-262-18253-X.

[55] Ioannis Rexakis and Michail G. Lagoudakis. Classifier-based policy representation. In *Proc. of the Int. Conf. on Machine Learning and Applications (ICMLA)*, pages 91–98, San Diego, CA, USA, 2008. doi:10.1109/ICMLA.2008.31.

[56] Martin Riedmiller and Heinrich Braun. A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In *Proc. of the IEEE Int. Conf. on Neural Networks (ICNN)*, pages 586–591, San Francisco, CA, USA, 1993. doi:10.1109/ICNN.1993.298623.

[57] Abel Rodríguez, David B. Dunson, and Alan E. Gelfand. The nested Dirichlet process. *Journal of the American Statistical Association*, 103(483):1131–1154, 2008. doi:10.1198/016214508000000553.

[58] Karsten Rohweder, Philipp Vorst, and Andreas Zell. Improved mapping of RFID tags by fusion with spatial structure. In *Proc. of the European Conf. on Mobile Robots (ECMR)*, 2009.

[59] Stuart Russell and Peter Norvig. *Artificial Intelligence – A Modern Approach*. Prentice Hall, 2nd edition, 2003. ISBN 978-0-13-790395-5.

[60] Sebastian Schneegans, Philipp Vorst, and Andreas Zell. Using RFID snapshots for mobile robot self-localization. In *Proc. of the European Conf. on Mobile Robots (ECMR)*, Freiburg, Germany, 2007.

[61] Paul Schnitzspan, Stefan Roth, and Bernt Schiele. Automatic discovery of meaningful object parts with latent CRFs. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 121–128, 2010. doi:10.1109/CVPR.2010.5540220.

[62] Anton Schwaighofer, Marian Grigoras, Volker Tresp, and Clemens Hoffmann. GPPS: a Gaussian process positioning system for cellular networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2003.

[63] Martin Senk. Vergleich von Suchstrategien für die Navigation in annotierten Karten. Bachelor's thesis, University of Freiburg, Germany, 2009.

[64] Vinay Seshadri, Gergely V. Záruba, and Manfred Huber. A Bayesian sampling approach to in-door localization of wireless devices using received signal strength indication. In *Proc. of the IEEE Int. Conf. on Pervasive Computing and Communications (PerCom)*, pages 75–84, 2005. ISBN 0-7695-2299-8. doi:10.1109/PERCOM.2005.1.

[65] Andrew Smith, Trevor Cohn, and Miles Osborne. Logarithmic opinion pools for conditional random fields. In *Proc. of the 43rd Annual Meeting of the Assoc. for Computational Linguistics (ACL)*, pages 18–25, 2005.

[66] Luciano Spinello, Rudolph Triebel, Dizan Vasquez, Kai O. Arras, and Roland Siegwart. Exploiting repetitive object patterns for model compression and completion. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2010. doi:10.1007/978-3-642-15555-0_22.

[67] Cyrill Stachniss, Giorgio Grisetti, and Wolfram Burgard. Information gain-based exploration using Rao-Blackwellized particle filters. In *Proc. of Robotics: Science and Systems (RSS)*, pages 65–72, Cambridge, MA, USA, 2005.

[68] Erik B. Sudderth, Antonio Torralba, William T. Freeman, and Alan S. Willsky. Describing visual scenes using transformed objects and parts. *Int. Journal of Computer Vision*, 77(1–3):291–330, 2008. doi:10.1007/s11263-007-0069-5.

[69] Yee Whye Teh and Dilan Görür. Indian buffet processes with power-law behavior. In *Advances in Neural Information Processing Systems (NIPS)*, 2009.

[70] Yee Whye Teh and Michael I. Jordan. Hierarchical Bayesian nonparametric models with applications. In Nils Lid Hjort, Chris Holmes, Peter Müller, and Stephen G. Walker, editors, *Bayesian Nonparametrics: Principles and Practice*. Cambridge University Press, 2010. ISBN 9780521513463. doi:10.1017/CBO9780511802478.006.

[71] Yee Whye Teh, Michael I. Jordan, Matthew J. Beal, and David M. Blei. Hierarchical Dirichlet processes. *Journal of the American Statistical Association*, 101(476):1566–1581, 2006. doi:10.1198/016214506000000302.

[72] Yee Whye Teh, Dilan Görür, and Zoubin Ghahramani. Stick-breaking construction for the Indian buffet process. In *Proc. of the Int. Conf. on Artificial Intelligence and Statistics (AISTATS)*, pages 556–563, 2007. URL http://jmlr.csail.mit.edu/proceedings/papers/v2/teh07a/teh07a.pdf.

[73] Romain Thibaux and Michael I. Jordan. Hierarchical beta processes and the Indian buffet process. In *Proc. of the Int. Conf. on Artificial Intelligence and Statistics (AISTATS)*, pages 564–571, 2007. URL http://jmlr.csail.mit.edu/proceedings/papers/v2/thibaux07a/thibaux07a.pdf.

[74] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics*. MIT Press, 2005.

[75] Philip A. Titus and Peter B. Everett. Consumer wayfinding tasks, strategies, and errors: An exploratory field study. *Psychology and Marketing*, 13(3):265–290, 1996. doi:10.1002/(SICI)1520-6793(199605)13:3<265::AID-MAR2>3.0.CO;2-A.

[76] Rudolph Triebel, Jiwon Shin, and Roland Siegwart. Segmentation and unsupervised part-based discovery of repetitive objects. In *Proc. of Robotics: Science and Systems (RSS)*, Zaragoza, Spain, 2010. URL http://roboticsproceedings.org/rss06/p09.pdf.

[77] K. E. Trummel and J. R. Weisinger. The complexity of the optimal searcher path problem. *Operations Research*, 34(2):324–327, 1986. doi:10.1287/opre.34.2.324.

[78] Peter Orbanz und Yee Whye Teh. Bayesian nonparametric models. In Claude Sammut and Geoffrey I. Webb, editors, *Encyclopedia of Machine Learning*, pages 81–89. Springer, 2010. doi:10.1007/978-0-387-30164-8_66.

[79] Shrihari Vasudevan and Roland Siegwart. Bayesian space conceptualization and place classification for semantic maps in mobile robotics. *Robotics and Autonomous Systems*, 56(6):522–537, 2008. doi:10.1016/j.robot.2008.03.005.

[80] Philipp Vorst and Andreas Zell. *European Robotics Symposium 2008*, volume 44/2008 of *Springer Tracts in Advanced Robotics*, chapter Semi-Autonomous Learning of an RFID Sensor Model for Mobile Robot Self-localization, pages 273–282. Springer, 2008. ISBN 978-3-540-78315-2. doi:10.1007/978-3-540-78317-6.

[81] Ingo Wegener. Optimal search with positive switch cost is NP-hard. *Information Processing Letters*, 21(1):49–52, 1985. doi:10.1016/0020-0190(85)90108-5.

[82] Brian Yamauchi. A frontier-based approach for autonomous exploration. In *Proc. of the IEEE Int. Symp. on Computational Intelligence in Robotics and Automation (CIRA)*, pages 146–151, 1997. doi:10.1109/CIRA.1997.613851.

[83] Hendrik Zender, Oscar Martínez Mozos, Patric Jensfelt, Geert-Jan M. Kruijff, and Wolfram Burgard. Conceptual spatial representations for indoor mobile robots. *Robotics and Autonomous Systems*, 56(6):493–502, 2008. doi:10.1016/j.robot.2008.03.007.