Information Theoretical Action Selection

by

Christian Potthast

A Dissertation Presented to the
FACULTY OF THE USC GRADUATE SCHOOL
UNIVERSITY OF SOUTHERN CALIFORNIA
In Partial Fulfillment of the
Requirements of the Degree
DOCTOR OF PHILOSOPHY
(Computer Science)

Committee:

Prof. Gaurav S. Sukhatme (Chair)	Computer Science
Prof. Stefan Schaal	Computer Science
Prof. Sandeep K. Gupta	Electrical Engineering

August, 2016

Dedicated to my family

Acknowledgements

First of all, I would like to thank my Ph.D. advisor Professor Gaurav Sukhatme for his guidance and supported during my time at USC. I especially thank Gaurav for giving me the freedom and confidence to pursue my own ideas. Diving into the large field of robotics was not easy if you can do whatever you want and need to somehow narrow it down, but it was immensely rewarding once you have found your passion and see ideas come to life on a robot. I could not have asked for a better advisor during this journey.

A special thanks also to all of my thesis proposal and defense committee members. Professor Stefan Schaal, Professor Fei Sha, Professor Hao Li and Professor Sandeep Gupta, thank you for all your support and guidance.

My first research experience, involving robots, was at the Technical University of Munich during my time as a Computer Science Diplom student. During this time, I worked in the group of Professor Michael Beetz under the direct supervision of Alexandra Kirsch, Armin Mller and Radu Bogdan Rusu. It was a great experience and laid the foundation to where I am now.

While still at the Technical University of Munich I spent over a year at the Georgia Institute of Technology, working with Professor Frank Dellaert. A very special thanks to Frank for giving me the opportunity and chance to work and learn from him. No doubt, without Frank I would have never made the decision of pursuing my Doctoral degree in the USA. During my time at GT I met some great people that all helped me in one way or the other with OCaml, SLAM, graph optimization and ease my way into the American way of life. To just name a view, Professor Henrik Christensen, Richard Roberts, Minxuan Sun, Michael Kaess, Sang Min Oh, Carlos Nieto, Alireza Fathi, Grant Schinlder, Kai Ni and Matthias Grundmann.

Besides working as a researcher at USC I was also fortunate to have spent time at two companies. My first internship was at Willow Garage, the creator of the PR2 robot. Here, I was again fortunate to be able to work with Radu Bogdan Rusu who gave me great ideas toward my thesis research work. During long nights and on the weekend I shared my office with Koen Buys, Bernhard Zeisl and Ryohei Ueda, thanks for the great time. My second internship was at the Toyota InfoTechnology Center. Here, I would like to thank Eric Martinson and Akin Sisbot for their guidance and support. With my office mate Young-Ho Kim I had great conversations and good times with our daily raid of the company fruit platter.

During my time as a graduate student, but especially in my very early years I had great support from labmates. Jonathan Kelly, Sameera Poduri, Karthik Dantu and Ryan Neal Smith, were great mentors and I am fortunate to have had them when I started. Great help also came from Geoff Hollinger, Jörg Müller and Andreas Breitenmoser. A

special big thanks to Jörg and Andreas, who were instrumental in helping me build a quadrotor platform and with my research during the final years of my thesis work.

During the days and sometimes during the nights I spent my time with my RESL peers in the lab or out in Los Angeles. Together we shared many ups and downs and I would like to thank everyone for many memorable moments. Thank you, Jon Binney, Jnaneshwar Das, Arvind Pereira, Hordur Heidarsson, Megha Gupta, Harshvardhan Vathsangam, Stephanie Kemna, Max Pflueger, David Inkyu Kim, Karol Hausman, Artem Molchanov, James Preiss, Oliver Kroemer and Lantao Liu. I would also like to thank Carl Oberg for his help and advice in equipping robots with sensors, helping with experimental setups and many interesting conversations.

I would also like to thank all the people from the Interaction Lab, CLMC, and ACT lab. You all made the time at USC unforgettable. Thank you, Ross Mead, Aaron St. Clair, David Feil-Seifer, Elizabeth Cha, Vince Enachescu, Bharath Sankaran, Harry Su, Mrinal Kalakrishnan, Peter Pastor, Ludovic Righetti, Jeannette Bohg, Wolfgang Hönig.

Finally, I would like to thank my parents Dr. Michael Potthast and Constanze Klimt as well as my brother Andreas Potthast for their unconditional support throughout the years. They always believed in me and encouraged me to find and go my own way. I am also very thankful that my grandpa Professor Dr. Dr. med. Ferdinant Klimt was still able to see me succeed in receiving my doctoral degree.

This has been an amazing journey and I am very fortunate and honored to have been able to pursue my PhD at USC. I would do it all over again!

Table of Contents

List of Figures

Abstract

For robots to one day become fully autonomous and assist us in our daily lives, they need to be able to self reliantly acquire information about their environment. Challenges arise from the limited energy budget to operate the robot, occlusion from the environment, and uncertainty in data captured by noisy sensors. To cope with such challenges, the robot needs to be able to rely on a system that enables it to capture information efficiently while respecting constraints. Furthermore, information acquisition should be reactive to sensor measurements, incorporate uncertainty and tradeoff information gain with energy usage. In this thesis we study such systems using information theoretical measures to formulate as general and versatile a framework as possible. Specifically, we examine the task of defining objective functions that enable us to tradeoff information with acquisition cost, enabling the robot to gather as much useful information as possible, but at the same time keep energy consumption to a minimum. We address this challenge for a variety of different tasks.

First, we examine the problem of 3D data acquisition, which is of outmost importance to a robotic system since the robot needs to know the environment it is operating in. We present an information gain-based variant of the next best view problem for an occluded environment. The developed framework enables the robot to quickly acquire information by sequentially choosing next observation positions that maximize information. Next, we look at adaptive action selection in the context of object recognition on robots with limited operating capabilities. We propose an information-theoretic framework that combines and unifies two common techniques: view planning for resolving ambiguities and occlusions and online feature selection for reducing computational costs. Concretely, this framework adaptively chooses two strategies: utilize simple-to-compute features that are the most informative for the recognition task or move to new viewpoints that optimally reduce the expected uncertainties on the identity of the object. Lastly, we present an online trajectory optimization approach that improves object recognition performance. With the idea in mind that the robot needs to make progress towards a goal, a cost function is formulated that allows the robot to improve recognition performance, maneuver safely through the environment and reduce information acquisition cost, while simultaneously moving towards the goal point. Through extensive experiments, in simulation and on real robotic systems, this thesis shows the usefulness of adaptive action selection. We have studied adaptive action selection on two concrete problems, but the use case and need for such an approach can be found in many other applications.

Chapter 1

Introduction

Currently, the use of autonomously operating robots in our daily live is still minimal. The few existing exceptions are robots that either operate in very controlled environments or are used for highly specialized tasks. We can think of endless applications and scenarios in which robots can enrich and ease our life, with the holy grail being a multi-purpose robot that can assist us in many different aspects of our life (Fig. 1.1). For autonomous operation, such robots need to understand the environment they operate in by acquiring information about its surrounding. In order to acquire information the robot needs to utilize its various sensors and process the sensor measurements to build an internal representation of the world it operates in. However, due to occlusions one observation is oftentimes not sufficient to fully capture the environment. To overcome this problem the robot needs to execute an action, this could be moving to another observation location, to be able to capture previous unseen parts of the environment and gain new information.

In general, the more information a robot has about its environment the more robustly and efficiently it can execute tasks. Thus, ideally the robot would collect as much information as possible, however this is often not possible because of resource or time constraints. For instance, mobile robots are often underpowered and need to be frugal with resources for computation, reasoning and moving to new positions. In settings like this, we want to keep the *cost* to a minimum, while at the same time acquire as much *useful* information as possible. Unfortunately, most environments a robot operates in are unknown to the robot, either it has never been observed before or the environment has changed compared to last time the robot has observed it. This has the consequence that we can not optimally precompute the actions necessary for the robot to acquire all information needed to complete a task. Instead, the robot needs to make decision online on how and what information needs to be capture or computed.

Figure 1.1: In the future we will have a variety of different robots inhabiting our home with us. We might have robots in our home that help us clean, cook, do our laundry and even be our personal butler.

Adaptive action selection is an appealing paradigm to overcome these challenges by trading off between the cost (computation, movement, etc.) and acquiring new useful information. Action selection is the process of selecting "what to do next" in unpredictable environments in real time, given an objective function that evaluates individual actions for their utility. It enables exploratory reactions to unforeseen events or scenes, and offers greater flexibility in actively seeking the most useful information for various information acquisition tasks. The general idea is for the sensors to be controlled such that only data with high information content is collected, and adapted dynamically to the robot's knowledge (i.e., belief state about the world) as a result of running inference on previously acquired information.

Action Selection

The process of selecting online "what to do next" in unpredictable environments.

A typical sequential action selection process, shown in Fig. 1.2, consisting of three modules: Action Set, Select Action and Action Execution. The main problem of the action selection process is complexity. Each considered action involves computations. Since all computations take time, processing resources and require memory, the robot can not possibly evaluate every action imaginable. Consequently, the robot must constrain its search to a subset of possible actions. This subset of all possible actions is defined in the Action Set. To find the best action of the set each individual action needs to be evaluated for its utility given a defined objective function. Intuitively, the objective function ranks the individual actions for their usefulness. Commonly, the action with the highest utility is picked as the next action. Finally, we execute this action and either repeat the previous described process or terminate the process determined by a termination criteria.

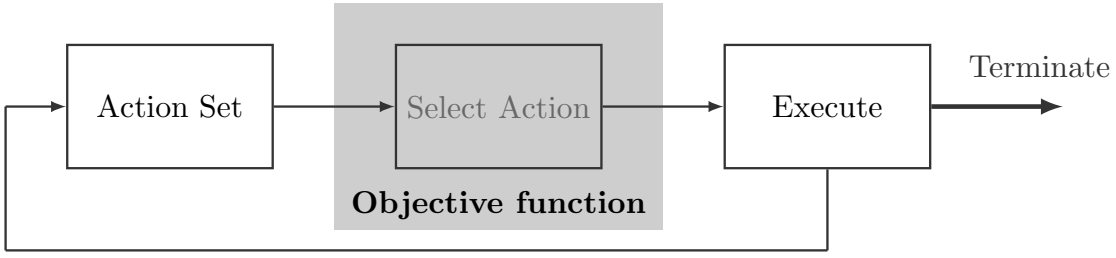


Figure 1.2: A typical sequential action selection process, consisting of three modules: Action Set, Select Action and Action Execution. First, a subset of possible actions is defined. Next, we evaluate the utility of all actions in the action set. Finally, we execute the action with the highest utility. This process repeats until we terminate given a termination criteria.

Concretely, the value of an action is estimated by the objective function and can be evaluated in many different ways, for example using a heuristic or information theoretical (Shannon, 1948) approach. Information theoretic approaches are popular in robotics, because it is based on probability theory and statistics. Using probabilistic models in robotic applications has tremendous advantages, since we can directly model the uncertainty in data captured by noisy sensors. There are many sources that introduce uncertainty, measurement noise is the most common one and in general dealt with by formulating an observation model. Others are much harder to deal with like uncertainty stemming from incomplete information. Both and many more can be modeled using probabilistic models, which me frequently use throughout this work. Consequently, using a formalism that can incorporate probabilistic estimations into our world belief state is of utmost value. Evaluating actions with information theoretic formulations allows us to do exactly that, quantifying actions in terms of information gain and uncertainty reduction given our current belief state.

Given the advantages of probabilistic reasoning in robotics and the ability to incorporate uncertainties into information theoretic quantities, we present in this work how they can be utilized for robotic applications. We have applied this for two important problems in robotics, *Autonomous 3D data acquisition* and *Object recognition*.

Autonomous 3D data acquisition and reconstruction of dense geometry of full environments or single objects is an often required task for robots. For this task, a substantial difficulty, and of most importance, is both the selection of viewpoints for data acquisition and the successful execution of subsequent tasks like reconstruction or object manipulation. Many algorithms have been developed to make this process autonomous and efficient for 2d (H. H. Gonzalez-Banos & Latombe, 2002) as well as 3D (Krainin, Curless, & Fox, 2011) environments. In general, the estimation of such viewpoints is known as the *Next Best View* (NBV) problem, first discussed in (Connolly, 1985). *Next Best View* approaches seek a single additional sensor placement or viewpoint given a heuristic or reward function. Since sensing actions are time consuming, one seeks a sequence of viewpoints which observe the environment in the minimum number of actions. Evaluating the utility of potential viewpoints is not easy, since it involves the prediction of the potential knowledge gain about the world. This is especially difficult in cluttered environments, because they introduce many visibility constraints due to occlusions. As a result, the difficulty of making accurate predictions of the knowledge gain is increased. In (Potthast & Sukhatme, 2014) we present an approach that incorporates prior knowledge about the environment to better estimate information gain in cluttered environments. Thus, more optimal observation positions are chosen which in turn reduces the number of actions needed

With 3D data acquisition only considering the sensor placement and their information gain, we next also consider the cost of processing the raw sensor data. In recent work (Potthast, Breitenmoser, Sha, & Sukhatme, 2015), we have proposed an adaptive action selection framework for object recognition on robots with limited operating capabilities in terms of energy. Energy is not just spent on viewpoint planning and motion of the robot, but also on processing of the raw sensor data into representative features. Conventional

approaches for object recognition are tour de force, relying almost exclusively on complex statistical models for classification and heavy engineering of computationally-intensive features (Bo, Ren, & Fox, 2011; Lowe, 2004). In stark contrast, our work hinges on the following intuition: *how to reduce overall costs for recognition by relying on very simple features extracted at multiple views in lieu of complex features extracted at a single view?*. Specifically, we describe an information-theoretic framework that unifies these two strategies: view planning for moving robots to the optimal view points and dynamic feature selection for extracting simple yet informative features. However, unlike standard feature selection, these two strategies are symbiotic in our algorithm. The selected features at any given time depend on the past trajectory of the view points while simultaneously affecting the future view point(s).

So far, the developed frameworks for both 3D data acquisition as well as for object recognition assumes information collection only when a next best observation position is reached. This means the robot is estimating a new observation location, moves to the next location and takes an observation. In practice, however this is highly inefficient since data could potentially, depending on the sensor, be collected while the robot is moving to a new observation location. To address this problem we present an online trajectory optimization approach that allows the robot to alter its trajectory such that the most useful information is collected while in motion. In this work, we formulate an objective function that optimizes a trajectory such that object recognition performance is improved. Inspired by prior work (Kalakrishnan, Chitta, Theodorou, Pastor, & Schaal, 2011), we formulate the optimization as a derivative-free stochastic optimization, allowing us to express the cost function in an arbitrary way. The cost function is defined such that information acquisition of target objects is improved, while simultaneously moving towards the goal point. We show the evaluation of our approach on a quadrotor platform in simulation as well as on a real robot. The results show that by using an online optimization approach recognition accuracy is greatly improved, but more importantly the optimized trajectory reduces the uncertainty of the object posterior class distribution greatly. Hence, verifying that the optimized trajectory collects more valuable information.

1.1 Contributions and Outline

The contributions of this thesis are made in the areas of information theoretical action selection for autonomous data acquisition, object recognition and feature selection as well as in trajectory optimization for online object recognition.

1. **Probabilistic Next Best View Estimation**

   We formulate an information-theoretic driven next best view framework for automatic data acquisition in cluttered environments. The key idea is to select new view points by iteratively estimating the knowledge gain of potential new observation positions. The approach reasons about the unknown space in cluttered environments by probabilistically estimating the likelihood of observing the unknown space.

2. **Active Multi-View Object Recognition and Online Feature Selection**

   We present an action selection approach for object recognition. We describe an information-theoretic framework that combines and unifies two common techniques: view planning for resolving ambiguities and occlusions and dynamic feature selection for reducing computational costs. Concretely, the algorithm adaptively chooses two strategies: utilize features that are the most informative to the recognition task, or move to new viewpoints that optimally reduces the expected uncertainties on the identity of the object.

3. **Online Trajectory Optimization to Improve Object Recognition**

   We present an online trajectory optimization approach that optimizes a trajectory such that object recognition performance is improved. Here, a quadrotor is tasked to follow a given trajectory, but to increase the recognition accuracy of potential target objects the system is able to alter the trajectory online. In this work we show that by optimizing the initial trajectory given a cost function, we can increase recognition accuracy and additionally show that data processing can be reduced by reducing the number of observations needed for reliable recognition.

The thesis is organized as follows. In Chapter 2, we introduce the information-theoretical formalism we use throughout this thesis. We choose an information theoretic approach because it is based on probability theory and statistics, enabling us to incorporate uncertainty into our selection of next actions. Chapter 3, we present an information-theoretic driven next best view framework for automatic data acquisition in cluttered environments. Next, in Chapter 4, an action selection approach for multi-view object recognition is presented. In Chapter 5 we present an online trajectory optimization approach that optimizes a trajectory such that object recognition performance is improved. Finally, in Chapter 6 we summarize the work presented in this thesis.

Chapter 2

Background

Throughout this work we will make extensive use of probability theory and information theory. Due to the nature of noise in sensors used on robots we need to deal with uncertainty while the robot is operating. Utilizing probabilistic models allows us to deal with such sensor noise and incorporate uncertainty in our computations. Furthermore, information theory, which is bases on probability theory and statistics allows us to quantify information. The most important quantities of information are entropy, the information in a random variable, and mutual information, the amount of information in common between two random variables. In the following we will give a short overview of the most important concepts and mechanism that allow us to quantify information using probabilistic methods.

2.1 Random Variable

A random variable $X : \Omega \to E$ is a measurable function from the set of possible outcomes Ω to some set E. When the range of X is finite or countable infinite, the random variable is called a discrete random variable and its distribution can be described by a probability mass function which assigns a probability to each value in the range of X. If the range is uncountable infinite then X is called a continuous random variable and takes values in $\mathbb{R}^d$ or in some smooth finite-dimensional manifold.

In this short introduction we will describe the concepts of information theory on the behavior of large sets of discrete random variables. A discrete random variable X is completely defined by the set of values is can take, Ω, which we assume to be a finite set, and its probability distribution $\{p(x)\}_{x\in\Omega}$. Here, the value $p(x)$ describes the probability that the random variable X takes the value x. The probability distribution $p : \Omega \to [0, 1]$ must satisfy the normalization condition

$$\sum_{x\in\Omega} p(x) = 1\,. \tag{2.1}$$

The expected value of a discrete random variable is the probability-weighted average of all possible values. In other words, each possible value the random variable can assume

is multiplied by its probability of occurring, and the resulting products are summed to produce the expected value. Formally we can define the expectation of the random variable X as follows:

$$E[X] = \sum_{i=1}^{\infty} x_i \, p(x_i) \, . \tag{2.2}$$

2.2 Information Theory

Information theory often concerns itself with measures of information of the distributions associated with random variables. Important quantities of information are entropy, a measure of information in a single random variable, and mutual information, a measure of information in common between two random variables. The relationships between some of the information theory measures, defined in the following subsections, are illustrated in Fig 2.1.

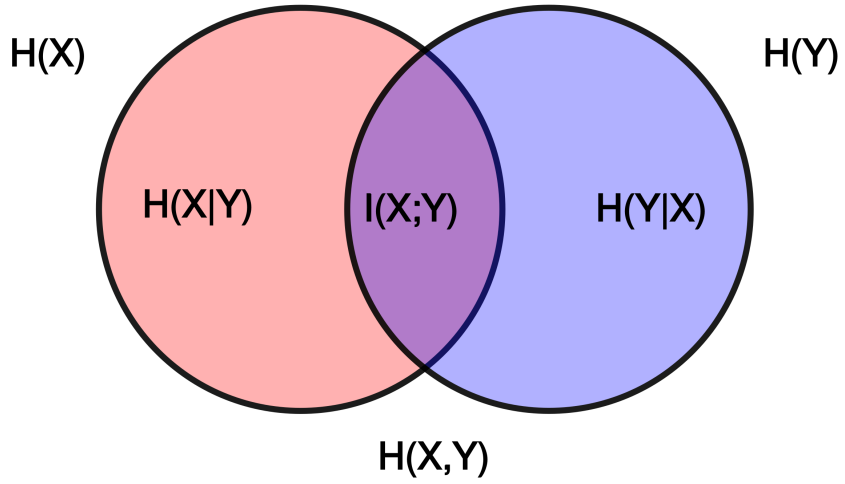


Figure 2.1: Spatial relationships between various entropy measures.

2.2.1 Entropy

Intuitively, the Shannon entropy gives a measure of the uncertainty of the random variable. It is a measure of surprise when looking at an outcome of the random variable. It is sometimes called the missing information: the larger the entropy, the less a priori information one has on the value of the random variable. The Shannon entropy H of a

discrete random variable X with possible values $\{x_1, \ldots, x_n\}$ and probability mass function $P(X)$ as:

$$H(X) = E[I(X)] = E[-\ln(P(X))]\,, \tag{2.3}$$

with E begin the expectation and I the information content of X. In the discrete case the Entropy can be computed as following:

$$H(X) = \sum_i p(x_i) I(x_i) = -\sum_i p(x_i) \log p(x_i)\,. \tag{2.4}$$

In the above formulation we use the logarithm to the base 2, which is well adapted to digital communication, and the entropy is then expressed in bits.

The entropy is zero if the outcome of the experiment is unambiguous and it reaches its maximum if all outcomes of the experiment are equally likely as shown in Fig. 2.2. Furthermore, the H is a strictly concave of X, i.e. for $0 < \lambda < 1$ and a random variable X and Y

$$H(\lambda X + (1-\lambda)Y) \geq \lambda H(X) + (1-\lambda) H(Y)\,, \tag{2.5}$$

with equality iff $\lambda = 0$, or $\lambda = 1$, or $X = Y$.‘

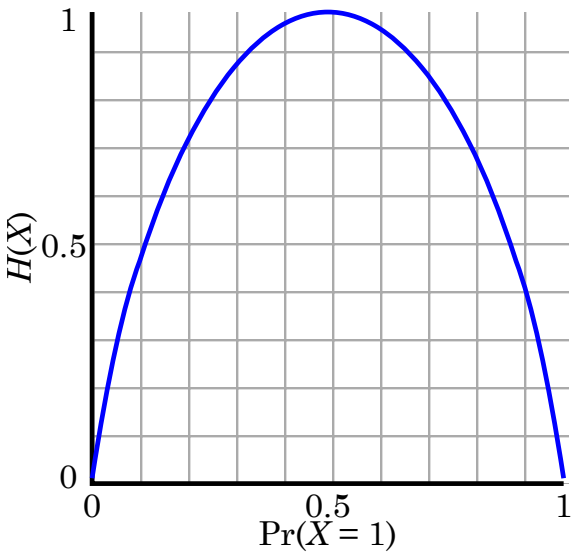


Figure 2.2: Entropy $H(X)$, measures the uncertainty in a random variable. We are most uncertain about the outcome if both values are equally likely.

2.2.2 Joint Entropy

The joint entropy of two discrete random variables X and Y is merely the entropy of their pairing: (X, Y). This implies that if X and Y are independent, then their joint entropy is the sum of their individual entropies.

$$H(X,Y) = E_{X,Y}[-\log p(x,y)] = -\sum_{x,y} p(x,y) \log p(x,y)\,. \tag{2.6}$$

2.2.3 Conditional Entropy

The conditional entropy quantifies the amount of randomness in the random variable X given that the value of another random variable Y is known. The entropy of X conditioned on Y is written as $H(X|Y)$ and defined as:

$$\begin{aligned} H(X|Y) &= E_Y[H(X|y)] \\ &= -\sum_{y\in Y} p(y) \sum_{x\in X} p(x|y) \log p(x|y) \\ &= -\sum_{x,y} p(x,y) \log \frac{p(x,y)}{p(y)}\,. \end{aligned} \tag{2.7}$$

2.2.4 Mutual Information:

Let X and Y be two random variables, $\mu_X(x)$ and $\mu_Y(y)$ the marginal probability density function of X and Y, respectively and $\mu_{X,Y}(x,y)$ the joint probability density function of (X,Y). Mutual Information expresses the reduction in uncertainty about variable X after we have observed Y:

$$\begin{aligned} I(X;Y) &= H(X) + H(Y) - H(X,Y) \\ &= H(X) - H(X|Y)\,. \end{aligned} \tag{2.8}$$

This means it measures how much knowing one of these variables reduces uncertainty about the other. From this notation we can write the Mutual Information as

$$\begin{aligned} I(X;Y) &= \sum_X \sum_Y \mu_{X,Y}(x,y) \log \frac{\mu_{X,Y}(x,y)}{\mu_X(x)\mu_Y(y)} \\ &= KL(P(X,Y) \parallel P(X)P(Y))\,. \end{aligned} \tag{2.9}$$

The mutual information is symmetric ,

$$I(X;Y) = I(;Y;X) \tag{2.10}$$

2.2.5 Relative Entropy

The relative entropy or *Kullback-Leibler divergence* between two probability distributions $P(X)$ and $Q(X)$ is defined as:

$$KL(P \parallel Q) = \sum_i p(x_i) \log \frac{p(x_i)}{q(x_i)} \,. \tag{2.11}$$

The relative entropy is a measure of "distance" between two probability distributions. The K-L divergence presents the amount of information lost if samples from the distribution P are assumed to be samples from Q.

However, $KL(P \parallel Q)$ is not a true "distance" since it is not symmetric and does not obey the triangle inequality. We can formulate a symmetric version base of the K-L divergence, also known as *Jeffrey's Divergence* as:

$$J(p \parallel q) = KL(p \parallel q) + KL(q \parallel p) \,. \tag{2.12}$$

Chapter 3

Probabilistic Next Best View Estimation

Major research areas in computer vision and robotics are autonomous 3D data acquisition and reconstruction of dense geometry of full environments or single objects. A substantial difficulty, and of most importance, is the selection of viewpoints for data acquisition and to accomplish subsequent tasks like reconstruction or object manipulation. Oftentimes we are limited by time, energy, or storage, but also occlusion making it necessary to estimate appropriate viewpoints for data acquisition. Over the past decade many algorithms have been developed to make this process autonomous and efficient for 2D (H. H. Gonzalez-Banos & Latombe, 2002) as well as 3D (Krainin et al., 2011) environments. A widely used technique for automatic data acquisition and exploration is to iteratively compute and place the sensor to a new observation position where a new sensing action is performed. From a pure reconstruction point of view, we face the problem of optimizing view selection from the set of available views. The data, from which we want to reconstruct, is usually presented as a video or as individual images, showing the world or object in question from different viewpoints (Haner & Heyden, 2012),(Dunn & Frahm, 2009). From this data, we want to find the subset of data that can best and most efficiently reconstruct the environment. In general, the estimation of such viewpoints known as the *Next Best View* (NBV) problem, first discussed in (Connolly, 1985), which seeks a single additional sensor placement or viewpoint given a heuristic or reward function.

In many cases sensing actions are time consuming, but also an increase in data does not guarantee a better reconstruction result, one seeks a sequence of viewpoints which observe the environment in the minimum number of actions or images. How many measurements are needed to observe the environment depends, in principle on the selected viewpoints and with that the gain in knowledge about the world. The potential knowledge increase of new observation positions can be estimated by reasoning about current world information. In practice however, making precise estimations of the knowledge increase is difficult and error prone, because the estimation involves predicting the unobserved portions of the world.

This is especially true in cluttered environments, because they impose many visibility constraints due to occlusion thereby increasing the difficulty of making accurate predictions of the knowledge gain. Inaccurate estimations lead to over or under prediction of the

gained information, hence possibly non-optimal observation positions are chosen. The more accurate we predict the information gain the fewer observation are necessary to fully observe the environment.

View planning approaches are used in a variety of different research ares. There are two major surveys of the problem, one describing the sensor planning problem in vision applications (K. a. Tarabanis, Allen, & Tsai, 1995). The other is a more recent survey of view planning for 3D object reconstruction and inspection (Scott, Roth, & Rivest, 2003). Of particular relevance to our work are approaches for exploration and automatic data acquisition using a robotic system. A survey for robot environment mapping and exploration can be found in (Thrun, 2002). In this brief review of related work, we focus on different approaches to estimate a set of potential new viewing positions as well as estimating the best position out of this set.

The main differences between existing approaches are the selection of the potential new viewing positions and the estimation of the NBV of this set. The earliest NBV estimation (Connolly, 1985) proposed two approaches to create a complete model of a scene. The new sensor position is chosen either by simulating viewpoints sampled from the surface of a sphere and scored by using ray-casting and change estimation in unknown space or by using a histogram of normals for unexplored octree node faces.

Several methods have been developed which generate the potential viewpoints uniformly randomly in the workspace. In (Nüchter, Surmann, & Hertzberg, 2003) an exploration algorithm was developed for a robot operating in 2D but acquired environment information in 3D. Potential new viewing positions are randomly generated on the 2D workspace of the robot and evaluated for their individual information gain. The potential new viewing positions are sampled uniformly randomly in a safe region (H. H. Gonzalez-Banos & Latombe, 2002), which consists of free space bounded by solid curves. To estimate the NBV position of this sample set a cost function trading off information gain and travel cost is evaluated and applied to each sample point. Similarly, in (Amigoni & Gallo, 2005) the potential viewing positions are sampled randomly from a list of line segments bordering unobserved space. However, compared to (H. H. Gonzalez-Banos & Latombe, 2002) the utility function which evaluates the viewing position is more complex and takes the precision of localization into account. This ensures better localization and registration of the individual measurement. In (Krainin et al., 2011) a probabilistic surface reconstruction approach is described using a depth cameras and a robotic manipulator holding an object. New viewing positions are uniformly sampled around the object and the manipulator hand. Given a current surface reconstruction a metric is formulated to score the potential new scanning positions in terms of reduction of uncertainty from virtual created depth maps.

Various methods are developed for reconstruction of dense geometry. In (Dunn & Frahm, 2009) the next best view estimation is used for efficient 3D model creation from images. The system estimates new viewpoints that best reduces the reconstructed model uncertainty. (Haner & Heyden, 2012) and (Hornung, Zeng, & Kobbelt, 2008) proposes a method for 3D reconstruction from a set of unordered images. The goal is to estimate a subset of images by sequentially estimating new viewpoint and select the best match.

Their approach reduces the computation time and increases accuracy of the reconstructed model. To find and select new viewpoints the authors in (Pito, 1999) and (Massios & Fisher, 1998) utilize surface normal information. Specifically, (Pito, 1999) introduces the notion of positional space, which allowed, in combination with a detailed scanner model, refraining of expensive ray-casting methods for the next best view estimation. In (Massios & Fisher, 1998) developed a quality constraint which estimated the quality of seen voxels in the scenes by utilizing the surface normal information. This improves the overall reconstruction of the measurement since viewpoints are selected which improve the laser range data quality of the scanned surface. The work of (Haner & Heyden, 2011) proposes a continuous optimization of method of finding many future next best view positions to reconstruct the captured environment. The system utilizes a real-time Visual simultaneous localization and mapping (SLAM) approach to estimate the sensor position and map the environment.

Camera in hand system used for model building are used by (Trummer, Munkelt, & Denzler, 2010) and (Wenhardt, Deutsch, Hornegger, Niemann, & Denzler, 2006). Both methods rely on images and feature tracking for measurement registration and model reconstruction. In (Trummer et al., 2010) the authors present a combined online approach of tracking, reconstruction and NBV estimation. In their work the extended e-criterion is used, which guids the sensor to the next position to minimize the reconstruction uncertainty. Feature tracking and 3D reconstruction in (Wenhardt et al., 2006) is done using a Kalman filter. The optimal NBV, which improves the reconstruction accuracy, is estimated by reducing the uncertainty in the state estimation.

A variety of frontier-based approaches have been developed that generate potential new viewing positions on the boundary between free and unobserved space. The earliest (Yamauchi, 1997) uses a probabilistic occupancy grid map to represent and reason about the environment. Adjacent frontier edge cells are grouped together into frontier regions. The centroids of these groups form the set of potential new viewing positions. The closest frontier region position is the NBV position. Frontier regions are also used in (Blaer & Allen, 2007) to build a detailed 3D outdoor map in a two step process. First a coarse exploration of the environment is performed by utilizing a 2D map and observability polygons to compute coverage positions. In the second step a 3D view planner samples potential view positions which lie on the frontier region. For each of the candidate viewpoints they count the number of visible cells using a ray-traversal algorithm. In the work of (Makarenko, Williams, Bourgault, & Durrant-Whyte, 2002) the authors build a complete framework and estimate the NBV position given localization quality, navigation cost and information gain. The information gain is computed over the entropy of the region around potential new NBV positions. (Stachniss & Burgard, 2003) introduces coverage maps to better represent the environment. They propose different methods to estimate the NBV position including trading of information gain and travel cost. Recent work by (Holz, Basilico, Amigoni, & Behnke, 2010) suggests improvements for continuous data acquisition for 2D mapping in confined spaces such as rooms. In their approach a room will be fully observed before a position with higher information gain is considered

as a candidate NBV position. Additionally, due to continuous data acquisition, NBV positions are constantly evaluated to avoid reaching a position which is already observed. In (Wettach & Berns, 2010) an exploration algorithm is developed for 3D data acquisition. Frontier cells, defined as known cells adjacent to unobserved cells, are ranked according their length of connecting grid cells. A subset of these are then ranked according their travel cost with the one with the minimal cost is chose as the NBV position.

Our approach is closest to (Krainin et al., 2011), (Blaer & Allen, 2007), (Stachniss & Burgard, 2003), (Makarenko et al., 2002) and (H. H. Gonzalez-Banos & Latombe, 2002) in terms of generating a suitable set of potential new viewing positions and evaluating the information gain. However, in contrast to previous work, we directly reason over the unknown space itself and estimate the expected knowledge gain from new views of cluttered environments. Previous proposed methods assume full visibility to the target region or object, which makes them poorly suited for cluttered environments since unseen objects in the occluded space lead to overestimation of the reward for potential new observation positions.

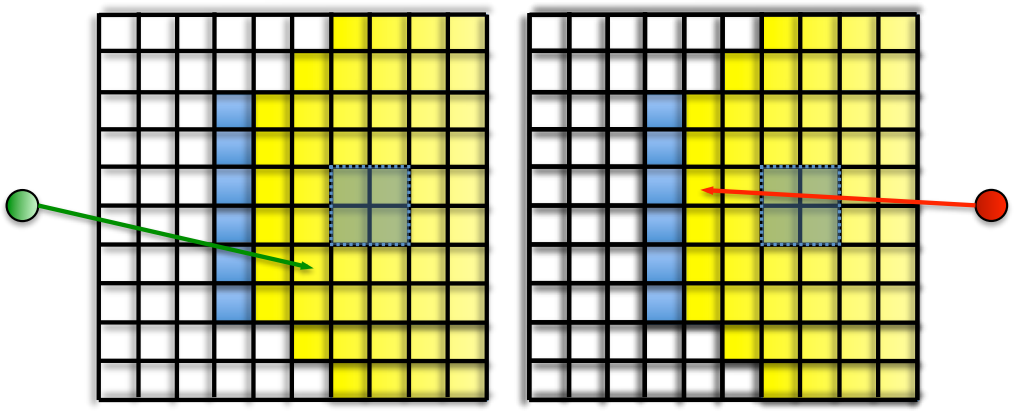
In the following sections, we describe a framework on automatic data acquisition and view point selection by iteratively estimating the knowledge gain of potential new observation positions. The information theoretical driven approach reasons about the unknown space in the cluttered environment by probabilistically estimating the likelihood of observing the unknown space. Intuitively, given the prior knowledge about the clutter in the scene, we try to model the fact that the deeper we look into unobserved regions, the higher the probability of observing an object we have not seen yet.

3.1 Next Best View estimation

In the NBV problem we are given the task to sequentially observe an environment by moving a sensor to new viewing positions. The process seeks to minimize some aspect of unobserved space (*e.g.* its size), though the exact form of the objective function will govern the precise optimization being sought.

At each stage of the process, unobserved regions of the environment are estimated using an occupancy grid and a ray-traversal algorithm, shown in Fig. 3.1a. Each cell c_n in the occupancy grid $\mathcal{O}$ is represented by a random variable $o_n \in [0,1]$ with $n = 1 \dots N$, where N is the total number of cells. The occupancy state $o_n = 0$ encodes the cell o_n as *free* and $o_n = 1$ as *occupied*. In order to estimate the NBV position many popular approaches (Stachniss & Burgard, 2003), (H. H. Gonzalez-Banos & Latombe, 2002), (Krainin et al., 2011), (Holz et al., 2010) formulate the problem in information theoretic terms and evaluate the gain in information of a set of potential new viewing positions $S = \{s_1, s_2, \dots, s_k, \dots, s_K\}$. The NBV position $\hat{s} \in S$ is then estimated by maximizing the expected information gain:

$$\hat{s} = \arg\max_k E[I(s_k)] \tag{3.1}$$



(a) (b)

Figure 3.1: The point cloud is represented as an occupancy grid. Blue cells are occupied space, white cells represent free space and yellow cells unknown space. Unknown space is estimated using a ray traversal algorithm shown in (a). To estimate the information gain of a potential next position we reason over the unknown space, as shown in (b). However objects which have not been seen yet (shaded square) lead to wrong estimations.

One information measure is entropy H, the amount of uncertainty in a discrete random variable. The entropy $H(o)$ expresses therefore the uncertainty about the state of a cell, in this setting whether the cell is free or occupied. For every potential new viewing position s_k, after integrating new measurements, the probability distribution of the occupancy grid either stays constant or changes. To measure the change we compute the information gain (Kullback-Leibler divergence) of the occupancy grid before and after integrating new measurements from a potential position s_k. Following the notation of (Thrun, Burgard, & Fox, 2005), the information gain I over the occupancy probability $p(o_n)$ of a given cell c_n is defined as:

$$I(p(o_n \,|\, s_k, z_{n,k})) = H(p(o_n)) - H'(p(o_n \,|\, s_k, z_{n,k})), \tag{3.2}$$

where $p(o_n \,|\, s_k, z_{n,k})$ is the occupancy probability after integrating a new measurement $z_{n,k}$ from location s_k according to our sensor model and where $H'(p(o_n \,|\, s_k, z_{n,k}))$ is the posterior entropy.

To estimate the posterior entropy we need to know what measurements will be measured if the sensor is placed at a certain position s_k. However, since this is not possible due to the unobserved portions in the occupancy grid, we have to integrate over all possible measurements to compute the expected information gain for a viewpoint s_k:

$$E[I(s_k)] = \int_z p(z \,|\, \mathcal{M}, s_k) \sum_{o_i \in C(s_k, z)} I(p(o_i \,|\, s_k, z_{i,k}))\, \mathrm{d}z \tag{3.3}$$

The set $C(s_k, z)$ defines the cells which are covered by the measurement z in the current occupancy grid map $\mathcal{M}$. Solving this integral in closed form is not feasible since we would have to integrate over all possible measurements.

A common approximation for the expected information gain is obtained by predicting the measurement z_k at position s_k using a ray-traversal algorithm. Typical methods to approximate the measurement include Markov Monte Carlo Sampling (Thrun et al., 2005) or assuming the measurement to be the first occurrence of a cell with high occupancy probability (Blaer & Allen, 2007), (Stachniss & Burgard, 2003), $p(o) > 0.5$. The latter approximation assumes infinite visibility until a cell with high occupancy probability is in line of sight. However, objects which have not yet been observed could potentially block the view to unobserved cells and therefore limit the actual information gain of a viewing position s_k. Thus, this could potentially lead to poor estimates over the unobserved space and the computed $\hat{s}$ is not necessarily the optimal choice, illustrated in Fig. 3.1b. The better we predict the outcome of a potential new observation position s_k the more accurate we can choose $\hat{s}$.

We propose to approximate the expected information gain by predicting how likely it is to see a specific cell from a position s_k. That is, we estimate the probability that cell c_n is observable from position s_k. In detail, we define $p(x)$ to be the observation probability of a cell $c \in \mathcal{O}$. Let $x_n \in [0, 1]$ be the random variable expressing the observation state, where $x_n = 0$ denotes the cell c_n is *not visible* and $x_n = 1$ denotes c_n is *visible*.

Given the observation probabilities we then estimate the expected information gain by estimating the posterior entropy over the occupancy state o of cell c, given x:

$$E[I(p(o \mid s, x, z))] \simeq H(p(o)) - H'(p(o \mid s, x, z)), \tag{3.4}$$

where z is the measurement predicted through a ray-traversal algorithm. We will now describe how the observation probabilities are computed in Section 3.1.1 and how they are used to estimate the posterior entropy is described in Section 3.1.2. Both of these subsections end with sample scenarios depicting the estimated distributions.

3.1.1 Observation Probability

In the following, the assignment of the observation probability $p(x_n)$ is formalized. The observation probability depends on all cells which lie in line of sight to the target cell c_n, estimated by a ray-traversal algorithm. Assume that the ray r to the target cell c_n penetrates M cells. Then r can be represented as the concatenation of penetrated cells $r = c_1 \dots c_m \dots c_M$. We want to estimate the observation probability of the target cell c_n given the cells in the ray. Intuitively, the observation probability of a cell $c_m \in r$ depends on the occupancy probability and the observability probability of the predecessor cell in the ray, $c_{m-1} \in r$. For instance, if the predecessor cell is highly likely to be occupied, the observation probability of c_m should be low. On the other hand, if the previous cell is likely to be free, chances are c_m is observable if the previous cell in the ray was also

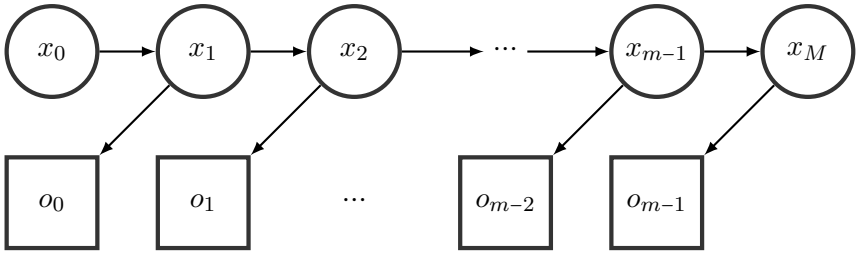


Figure 3.2: The estimation process modeled as a Hidden Markov Model. The observation state x_m is estimated from state x_{m-1} given all prior occupancy probabilities up to o_{m-2}.

likely to be observable. Thus we assume the observation probability to be Markovian and formulate the estimation process as a Hidden Markov Model, depicted in Fig. 3.2.

To estimate the observation state x_m we need to estimate the probability distribution associated with the current state conditioned on the measurements up to the state $m-2$. We can write the estimation of the probability distribution as a two step prediction-update recursion.

The prediction step, predicts the observation probability x_m from state x_{m-1} given all prior occupancy probabilities up to o_{m-2}:

$$p(x_m \,|\, o_{0:m-2}) = \sum_{x_m=\{0,1\}} p(x_m \,|\, x_{m-1})\, p(x_{m-1} \,|\, o_{0:m-2}), \tag{3.5}$$

where $p(x_m \,|\, x_{m-1})$ is the probability of transitioning from state x_{m-1} to x_m is. The state transition is the only open parameter in our system and controls the behavior of our approach, explained in more detail in section (3.1.1). The prior observation probability of the first cell in the ray r_1 is set to $x_0 = 1.0$, stating full observability of the first cell $c_0 \in r$.

The update step to the cell c_m integrates the occupancy state of cell c_{m-1} and is defined as

$$p(x_m \,|\, o_{0:m-1}) = \eta\, p(o_{m-1} \,|\, x_m)\, p(x_m \,|\, o_{0:m-2}), \tag{3.6}$$

with η as the normalization factor. Furthermore the measurement $p(o_{m-1} \,|\, x_m)$ can be written in terms of the current occupancy state of cell o_{m-1}

$$p(o_{m-1} \,|\, x_m) \propto p(x_m \,|\, o_{m-1}) = 1 - p(o_{m-1}). \tag{3.7}$$

Visibility State Transition Probability

The observation probability of a cell depends on the occupancy probabilities of the cells in the path of the ray cast as well as the state transition probability of the Markov Process. The transition probability defines the likelihood of state transitions given their probability values. However, the transition probability also controls the behavior of the Markov process and with that the resulting observation probability of a cell. Hence, choosing different

parameters for the state transition probabilities result in different visibility assumptions. This allows us to control the behavior of the observation probability when intersecting unobserved cells. As a result different parameters result in different NBV positions.

In the following we define the desired behavior of how cells in the ray should affect the target cell observation probability. Depending on the occupancy probability of the cells in the ray we want the observation probability of the target cell c_n to change in a certain way, when observed from a certain position s_n. If all cells in the ray r are likely to be free, we want the observation probability of cell c_n to be high (with high probability the cell c_n is observable). If the ray contains a 'likely occupied cell' at position c_m, we want the observation probability of the consecutive cell c_{m+1} to drop off steeply to a low observation probability value. Furthermore, we want the observation probability of all consecutive cells after c_m to be low and with that assign the target cell c_n a low observation probability. If the ray contains consecutive unobserved cells we want the observation probability to gradually fall. Assume the occupancy probability for the cell $\{c_0, c_1, \dots, c_{n-1}\}$ is uncertain. Then we want the observation probability follow the rule:

$$p(x_1) > p(x_2) > \cdots > p(x_{n-1}) > p(x_n) \tag{3.8}$$

The more uncertain cells the ray needs to penetrate, the smaller is the probability of that target cell c_n being visible. Intuitively this expresses the fact that the more unobserved space a ray penetrates to get to cell c_n, the more likely it is that the ray will be blocked by an object that has not yet been observed.

To achieve this we make the state transition probability $p(x_{n+1} \,|\, x_n)$ to change adaptively depending on how far into the ray we are. Let the transition matrix be of the form

$$p(x_{n+1} \,|\, x_n) = \begin{array}{c} \\ vis \\ \neg vis \end{array} \begin{array}{c} \begin{array}{cc} vis & \neg vis \end{array} \\ \begin{bmatrix} t & 1-t \\ 0.1 & 0.9 \end{bmatrix} \end{array}. \tag{3.9}$$

Then the variable t controls how likely it is to stay in the state *visible* and with that how likely it is to transition to the state *not visible*. The farther we get into the ray, the less likely we want it to be to stay in state *visible*. Thus we make t dependent on the number of cells passed N_{cells} so far:

$$t = a^{N_{cells}}, \tag{3.10}$$

where a controls how steeply the value of t falls off.

Observability Probability Scenarios

Examples of how the observability probability evolves as we pass through the ray can be seen in Fig. 3.3. In the upper half of Fig. 3.3 we can see the evolution of the observation probability. This shows how the probability changes when the ray passes through the

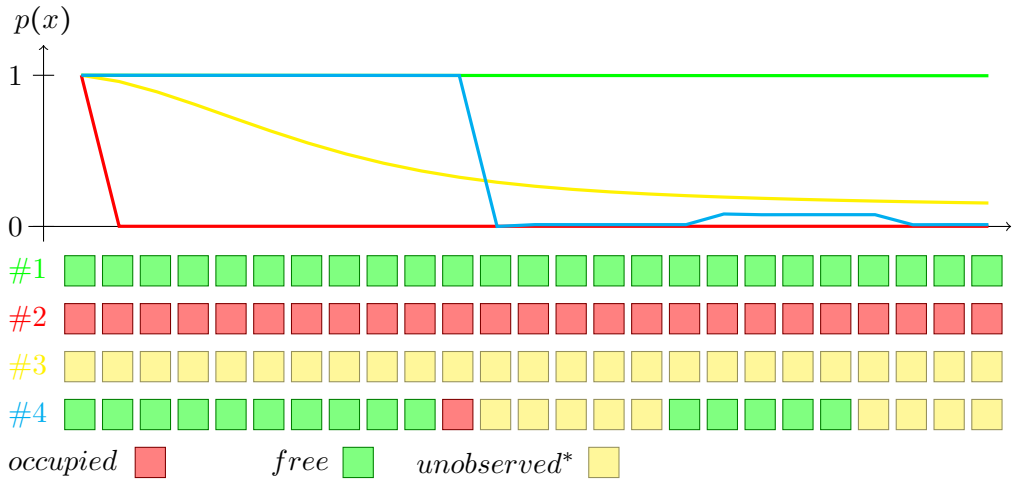


Figure 3.3: The figure shows the change in observation probability when applied to our model. The upper half of the picture shows the evolution of the observation probability when traversing the cells in a ray. The lower part of the figure show the states of the cell in the ray.

cells till it hits the target cell. In the lower part of the figure we can see four examples cases of what cells a ray might contain. Four different scenarios are shown, #1 where the ray-traversal algorithm had determined only free cells, #2 where only occupied cells where detected, #3 where only unobserved* cells where detected and finally #4 where a mix of all possible ray measurements where detected. In scenario #1 the observability probability stays almost constant close to $p(x) = 1$. This makes sense, since we do not pass any *occupied* or *unobserved* cells. In contrast, scenario #2 shows how $p(x)$ instantly drops to a value close to $p(x) = 0$ as soon as the first occupied cell is passed, and stays there since all new measurements are also occupied cells. The scenario #3 shows exactly the behavior that we have previously described. The ray consists of unobserved cells only, and the farther we get into the ray the more drops the observability probability $p(x)$. The last scenario #4 depicts the other wished for behavior, that is as soon as an occupied cell is passed, $p(x)$ drops and stays relatively small after that.

To better illustrate the cases and why the changes in probability occur in Fig. 3.3 and 3.5 a virtual scenes is shown in Fig. 3.4. The sub figures correspond to the cases displayed in Fig. 3.3 and 3.5 (#1 is 3.4a , #2 is 3.4b, #3 is 3.4c, #4 is 3.4d). Each cell in the grid is assigned a new observation probability, which depends on the cells traversed by the ray. The ray (blue error) is shot from the current sensor location (blue circle) to the target cell, with the white dashed boxes indicating all cell which are penetrated by the ray. The case #1 shown in Fig. 3.4a shows that the ray only traverses free cells. This

*Note, a cell is *unobserved* if its initial occupancy probability has not changed, meaning no measurement has been received for this cell yet. See Section 3.2.3 for more details.

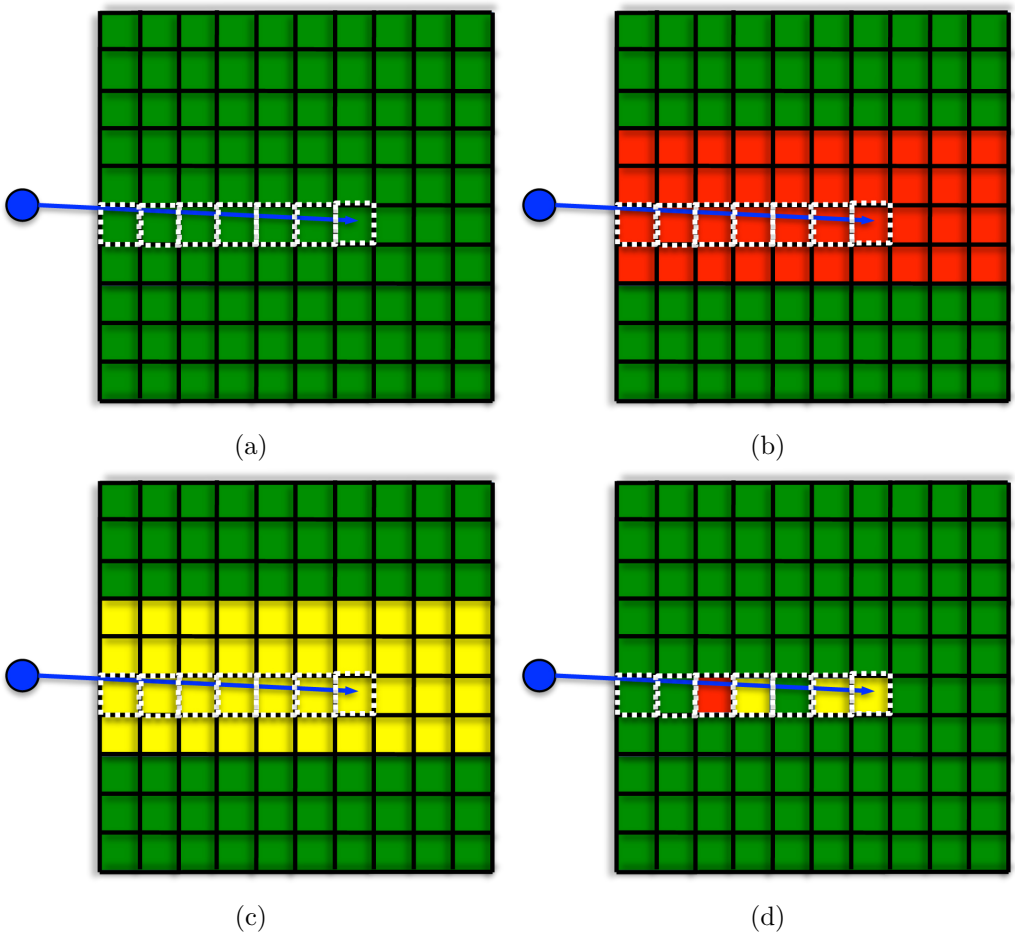


Figure 3.4: In this figure we can see virtual scenes corresponding to the cases presented in Fig. 3.3 and Fig. 3.5. The blue circle on the left of all figures is the current sensor location, the arrow represents one ray shot into the grid to a target cell and the dashed white cells represent the cells penetrated by the ray. Green cells are free, red are occupied, and yellow cells are unobserved. In 3.4a only free cells are penetrated, in 3.4b only occupied cells, in 3.4c only unobserved cells and in 3.4d a mix of all are possible states are penetrated.

occurs if we have estimated the cells in a previous observation as fee. In case #2 shown in Fig. 3.4b we traverse only occupied cells. This results when we have observed an obstacle so the cells occupied by the obstacle are occluded. In Fig. 3.4b which is shown in case #3 we traverse only unobserved cells. This occurs if the robot is positioned on the boundary of observed and unobserved space. In this case we have no knowledge about the environment and the probability behaves as shown in Fig. 3.3. In the last case #4 shown in Fig. 3.4d we traverse cells with different states. As we can see in Fig. 3.4d the ray first traverses free cells, followed by an occupied cell, followed by unobserved cell and so

on. This occurs if we have partially estimated the probabilities of the cells but some of them are still unobserved. Since we penetrate an occupied cells, belonging to an observed object, the target observation probability immediately decreases to zero, since we can not observe the cell.

3.1.2 Posterior Occupancy Probabilities

So far we have explained how to compute the observability probabilities of cells. Now we describe how this observability of a cell is used to estimate the posterior occupancy probability $p(o_m \,|x_m, z_m)$. Intuitively, the occupancy probability now depends on two factors: how visible is cell c_m, which is captured by the observability state x_m and what is the occupancy state of the predecessor cell c_{m-1} in the ray r. Thus, we can again use a Bayes Filter to compute the posterior occupancy probability $p(o_m|x_m, z_{1:m})$, through a recursive prediction-update process. The prediction step is given through:

$$p(o_m \,|\, x_m, z_{1:m-1}) = \sum_{o_{m-1}=\{0,1\}} p(o_m \,|\, o_{m-1}, x_m)\, p(o_{m-1} \,|\, z_{1:m-1}), \tag{3.11}$$

where $z_{1:m}$ are again the measurements predicted through the ray-traversal algorithm. Note, how the transition matrix depends on the visibility state x_m of the next cell c_m. Intuitively, the visibility of a cell c_{m-1} should influence how certain the transition to the next state is. We will explain this in more depth in Section 3.1.2. The measurement update takes the usual form of

$$p(o_m \,|\, x_m, z_{1:m}) = \eta\, p(z_m \,|\, o_m)\, p(o_m \,|\, x_m, z_{1:m-1}), \tag{3.12}$$

where η normalizes the distribution.

Occupancy State Transition Matrix

The intuition of why the occupancy transition matrix should depend on the obervability of the cell c_n becomes clear when we think about what effect the following transition matrix would have:

$$p(o_m \,|\, o_{m-1}) = \begin{array}{c} \\ occ \\ free \end{array} \begin{array}{c} \begin{array}{cc} occ & free \end{array} \\ \begin{bmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \end{bmatrix} \end{array}. \tag{3.13}$$

When using this transition matrix in the prediction step we will get a very uncertain belief $p(o_m \,|\, x_m, z_{1:m-1})$, even if the current occupancy state distribution $p(o_{m-1} \,|\, x_{m-1}, z_{1:m-1})$ was very certain. We would want this behavior if the observability of cell c_m is very low, because we want the occupancy distribution over c_m to become less certain. Only if cell c_m can be given a certain measurement update, the occupancy distributions becomes certain

again. Thus we define the transition matrix depending on the observability probability as follows:

$$p(o_m \,|\, o_{m-1}) = \begin{array}{c} \\ occ \\ free \end{array} \begin{array}{c} \begin{array}{cc} occ & free \end{array} \\ \begin{bmatrix} t & 1-t \\ 1-t & t \end{bmatrix} \end{array}, \tag{3.14}$$

where

$$t = 0.5 + \frac{p(x_m)}{2}. \tag{3.15}$$

Now, if cell c_m is likely to be visible the transition matrix is closer to the identity matrix and will not affect the posterior distribution $p(o_m \,|\, x_m, z_{1:m})$ by much. However, if cell c_m is likely to be not visible, the transition matrix will result in a less certain posterior distribution, unless a measurement with high certainty can be integrated.

Occupancy Probability Scenarios

We use the same scenarios from the observability estimation, to depict the behavior of the occupancy probability in Fig. 3.5 and illustrated as a virtual scene in Fig. 3.4. Again we can see how the resulting occupancy probabilities follow our intuition. Scenario #1 shows how the occupancy probability stays close $p(o) = 0$ as we pass through a ray full of free cells. Again in contrast, scenario #2 shows the opposite behavior, where the occupancy probability stays close to $p(o) = 1$ as we pass only occupied cells in the ray. In scenario #3 $p(o)$ converges against the most uncertain state of $p(o) = 0.5$ when only unobserved cells are in the ray. Finally, scenario #4 depicts, how the occupancy probability becomes close to $p(o) = 1$ as soon as we observe an occupied cell, but then drops again to the uncertain state of $p(o) = 0.5$ after that when encountering unobserved cells, and only dropping to almost $p(o) = 0.0$ when measurements of free cells come in.

3.2 Preprocessing

In this section we describe the general framework of our approach. First we give an overview of the system architecture followed by a detailed description of the individual steps.

3.2.1 System Architecture

Given a new point cloud $\mathcal{P}$, the first step is to perform the merging of two point clouds by transforming the new point cloud into the coordinate frame of the previous one. Let $\mathcal{P}_{n-1}$ denote the point cloud after $n-1$ cycles. Given the new point cloud $\mathcal{P}$, we join $\mathcal{P}_{n-1}$ with

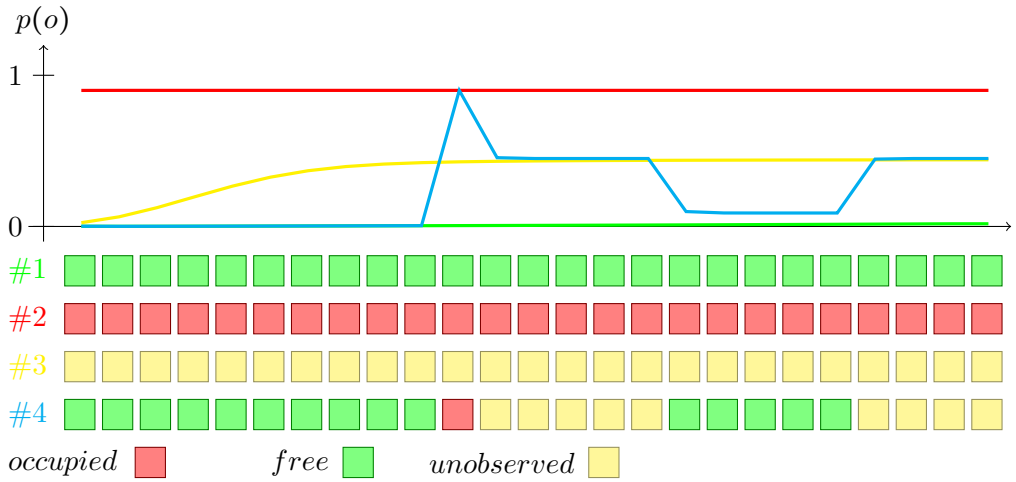


Figure 3.5: The occupancy probability is shown for each cell in a ray. The upper half of the picture shows the evolution of the occupancy probability when traversing the cells in a ray. The lower part of the figure show the states of the cell in the ray.

$\mathcal{P}$ to obtain $\mathcal{P}_n$. To merge two point clouds we need to find the rigid transformation of $\mathcal{P}$'s coordinate frame to $\mathcal{P}_{n-1}$'s coordinate frame. This transformation can be calculated using the current sensor pose and the first sensor observation pose. Using the current pose however leads to erroneous transformation matrices, due to inaccuracy in the sensor pose estimation. To improve the transformation we use an *Iterative Closest Point* (ICP) based approach. ICP is initialized with the estimated transformation using the current sensor pose and iteratively improves this initial transformation using information from $\mathcal{P}_n$ and $\mathcal{P}$. This procedure is explained in more detail in section 3.2.2.

For further processing the point cloud $\mathcal{P}_n$ is converted into an occupancy grid. However, the sparse representation of the point cloud leads to holes in the occupancy grid representation of the scene. To fill these holes we use a Markov Random Field (MRF) in combination with iterated conditional modes (ICM) (Bishop, 2006). MRFs are undirected graphs with every node conditioned only on its neighboring nodes. ICM is a greedy method to sequentially maximize local conditional probabilities. Eliminating such holes leads to improved estimation of occluded space when using a ray-traversal algorithm for occlusion estimation. The MRF model optimization is described in section 3.2.4.

In the next step we estimate the occluded space in the occupancy grid representation by using a ray-traversal algorithm. This also allows us to update the occupancy probabilities of every cell in the occupancy grid given our sensor model. The occupancy probability is either changing to 'likely free' or remains at uncertain. The definition of the occupancy grid as well as the ray-traversal algorithm used for estimating the occluded space is explained in section 3.2.3.

The final step is to determine the next best viewing pose by choosing the pose that yields the highest expected information gain as explained in Section 3.1.

3.2.2 Point Cloud Registration

In practice, estimation of the current sensor pose will become inaccurate over time. When transforming the point clouds based on erroneous localization, the clouds will be affected by the same drift.

To deal with this problem we use ICP (Besl & McKay, 1992) to find the transformation that minimizes the difference between two point clouds. ICP is designed to fit points in a target point cloud to points in a control point cloud. For ICP to work we need an initial transformation to align the point clouds coarsely before ICP is applied. The erroneous rigid transformation, obtained through the current poses estimation, is used as the initial transformation.

3.2.3 Occupancy Grid

To estimate and represent unknown space we use an occupancy grid, which is a discretized probabilistic representation of the environment. This makes it possible to assign every position in space a probability representing knowledge of the world and use efficient ray-traversal algorithm to estimate the unobserved space.

Every cell $c_{x,y,z}$ in the occupancy grid $\mathcal{O}$ is assigned a continuous random variable $p(o)$ denoting the occupancy probability of the cell. A probability of $p(o) = 1.0$ stating the cell is occupied while a probability of $p(o) = 0.0$ as free. An unobserved cell state is expressed with $p(o) = 0.5$.

Initially, all cells $c_{x,y,z} \in \mathcal{O}$ are initialized with probability 0.5, stating no knowledge about the state of any of the cells. For every new measurement $p \in \mathcal{P}$ we want to update the corresponding cell $c_{x,y,z} \in \mathcal{O}$ with a new occupancy probability. We update the occupancy grid using a stochastic sensor model defined by a probability density function $p(m|z)$, with m being the actual measurement and z being the true parameter state space value. In our case z corresponds to the cartesian coordinate of the cell center. We assume the probability function $p(m|z)$ to be Gaussian.

To determine the cell occupancy probability we incrementally update the occupancy grid using a sequential update formulation of the Bayes' theorem (Elfes, 1989) . Given a current estimation of the occupancy probability $p(o)$, we want to estimate $p(c_{x,y,z}|m_{x,y,z})$, given a new measurement m. The update is formulated with

$$p(p(o)\,|\,m_{t+1}) = \frac{p(m_{t+1}\,|\,p(o))\;p(p(o)\,|\,m_t)}{\sum\limits_{p(o)} p(m_{t+1}\,|\,z)\;p(z\,|\,m_t)} \tag{3.16}$$

Furthermore, let $\mathcal{U}$ be the set of all cells which have a probability of $c_{x,y,z} = p(o) > 0.4 \wedge p(o) < 0.6 \in \mathcal{O}$.

Unknown space estimation

Point Cloud data structures contain no information about the space where no measurements have been returned, consequently there is no distinction between space which is free and space which could not be measured due to occlusion. The distinction between the two types of space is important, because observing unobserved space gives us new scene information. With the use of a ray-traversal algorithm we distinguish between observed free space and occluded unobserved space. The ray-traversal algorithm we are using to estimate the new occupancy probability of the cells is describe in (Amanatides & Woo, 1987).

To estimate the occluded part of the scene we shoot rays from position P to every uncertain cell $c_{x,y,z} \in \mathcal{U}$. P is the position of the sensor from which it has acquired the input point cloud. The new probability of the cell depends on the ray ending in the cell $u_{x,y,z} \in \mathcal{U}$. Every ray r contains all cells it penetrates until it hits the target cell. If the ray contains a cell with probability $p(o_{x,y,z}) > 0.6$ (meaning the ray has penetrated a cell which is occupied) the target cell probability will remain the same. If the ray contains only free cells with $p(o_{x,y,z}) < 0.4$ or unobserved cells we change the probability of the cell according to our sensor model. This changes the occupancy probability of the cell towards more likely to be free. The change of the probability distribution is justified because P_i was the actual acquisition position and we can be certain that we see the unobserved cell if we penetrate only likely to be free or unobserved cells. Otherwise we would have measured an occupied cell and therefore penetrated an occupied cell.

Cost

Depending on the occupancy grid dimensions the number of cells can be very large. The cost for estimating the unobserved space from an acquisition position P depends on the number of unobserved cells $\mathcal{U}$ times the number of cells in each ray. Initially, the set $\mathcal{U}$ is fairly large since $\mathcal{O}$ is initialized with all cell probabilities set to 0.5 minus the cells that we have received measurements for and set to most likely to being occupied. However the number of unobserved cells drastically decreases with every new data acquisition, hence the cost for reestimating the unobserved space after incorporating new measurements also decreases.

3.2.4 Markov Random Field

The sparseness of point cloud can lead to holes in the surfaces of objects in the environment when converting them into an occupancy grid, resulting in cells that should be occupied remain unobserved. This phenomenon can be seen in Fig. 3.6a and 3.6c, showing a partially observed box and cylinder. Holes in the objects lead to incorrect estimation of the unobserved space. Thus we want to identify the unobserved cells which belong to an object and set their probability to occupied. Intuitively, cells that are surrounded by occupied cells are likely to be part of the object as well. These correlations between neighboring cells can be captured using a Markov Random Field (MRF), were each variable only depends

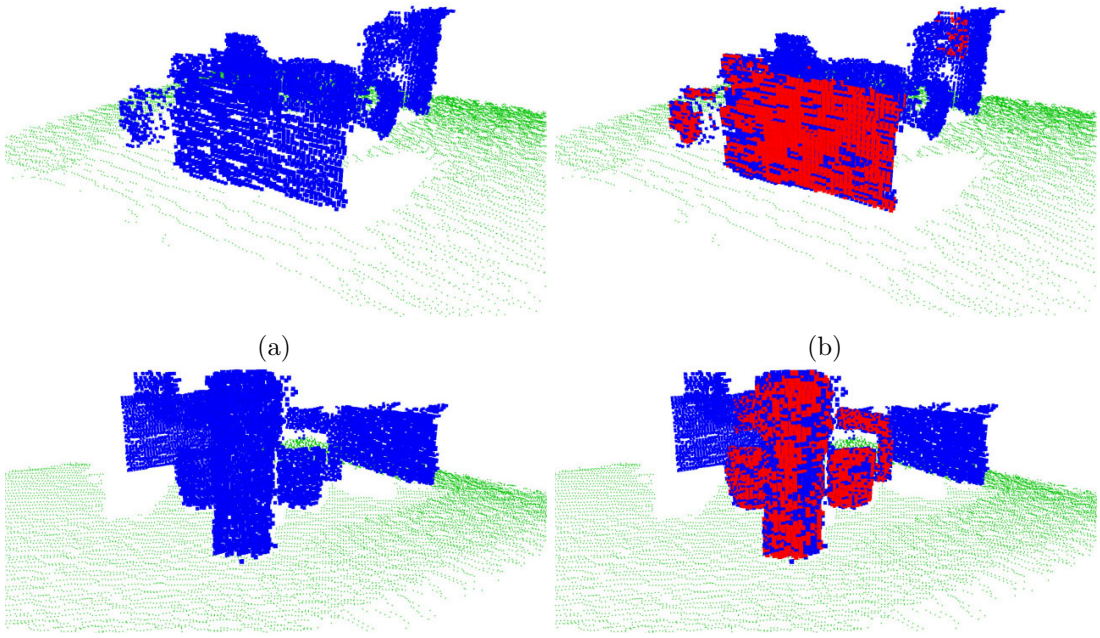


Figure 3.6: In (a) and (c) we can see missing data of the box and a cylinder. This missing measurements lead to holes in the objects. The result of the MRF approach can be seen in (b) and (d) where the red cells are the cells set to occupied after convergence of the MRF.

on the neighboring variables. In a $1D$ MRF the probability value of a random variable X_n depends on its two neighbors

$$P(X_n|X_{n-1}, X_{n+1}). \tag{3.17}$$

With the point cloud situated in $\mathbb{R}^3$ the MRF in a 3-D setting has $S = \{s_{x,y,z}\}$ which is the set of hidden nodes and $O = \{o_{x,y,z}\}$ the set of observation nodes, where $x = 1 \dots X, y = 1 \dots Y, z = 1 \dots Z$, with X, Y, Z being the dimensions of the occupancy grid . The nodes in S are related to one another over a neighboring system. For a fixed site s we define a neighborhood $N(s)$. To make it more concrete for the site $s_{x,y,z}$ the neighborhood is defined as: $N(s_{x,y,z}) = \{(x-1, y, z), (x+1, y, z), (x, y-1, z), (x, y+1, z), (x, y, z-1), (x, y, z+1)\}$. For efficiency we are not considering the diagonals. Thus in a 3D setting, the probability of a node s depends on its neighborhood,

$$P(s|N(s)) \tag{3.18}$$

Both the observation variables $o_{x,y,z} \in \{-1, +1\}$ and the hidden variables $s_{x,y,z} \in \{-1, +1\}$ are initialized with the state of the corresponding cell, where +1 represents an occupied cell and −1 the state unobserved. Furthermore the structure of our graphical model can

be summarized by presence of two types of cliques. The first group of cliques are formed by $\{s_{x,y,z}, o_{x,y,z}\}$ which have an associated energy function modeling the relationship of the observation variable and the hidden variable. We choose an energy function which favors equal signs of $\{s_{x,y,z}\}$ and $\{o_{x,y,z}\}$ by giving a lower energy if both have the same sign and a higher energy when they have the opposite sign

$$E(s_{x,y,z}, o_{x,y,z}) = -\eta \cdot s_{x,y,z} \cdot o_{x,y,z}, \tag{3.19}$$

where η is a positive constant. The other group of cliques are formed by pairs of neighboring hidden variables, $s_{x,y,z}$ and $z_n \in N(s_{x,y,z})$. Here, we also want the energy to be low if both cells have the same sign and a high energy if they are different. We define the energy function as follows

$$E(s_{x,y,z}, z_n) = -\beta \cdot x_{x,y,z} \cdot z_n, \tag{3.20}$$

with β being a positive constant. Thus, the total energy function of our model is given by:

$$E(S, O) = -\beta \sum_{s \in S} \prod_{z_n \in N(s)} s \cdot z_n - \eta \sum_{x,y,z} s_{x,y,z} \cdot o_{x,y,z} \tag{3.21}$$

To maximize the joint distribution of the MRF we use the *iterative conditional modes* (ICM) (Tilton & Swain, 1984) algorithm, which "greedily" maximizes local conditional probabilities sequentially. The optimal assignment of the hidden variables $s \in S$ is found by iterating over the field, taking always one node variable $s_{x,y,z}$ at a time and evaluating the total energy for the two possible states $s_{x,y,z} = +1$ and $s_{x,y,z} = -1$ while keeping all other node variables fixed. After calculating the total energy for both states, we assign s the state that produced the lower energy. It should be noted, that we only iterate over hidden variables $s \in S$ that where initialized with state unknown. This is done for two reasons: On the one hand, we are certain about the assignment of a cell being occupied, thus we do not want the state of hidden variables, that were initialized with $s_{x,y,z} = o_{x,y,z} = +1$, to change. On the other hand, this leads to faster convergence of the algorithm. The algorithm has converged if we have visited every site $s \in S$ at least once and no changes to the variables have been made. A typical result of the MRF optimization step is shown in Fig. 3.6b and 3.6d.

3.3 Experiments

We evaluate our formulation of the NBV problem in the task of large scale exploration and detailed data acquisition using a robotic system. The two evaluation scenarios differ in their size and level of clutter. To evaluate the detailed data acquisition capabilities we use the robotic system to explore a table top environment and acquire object information. The table top scene is extremely cluttered and unstructured resulting in many occlusions and the need to observe the table top from different positions. This environment analyzes the algorithm in the presence of clutter. In the second experiment, which is conducted only in simulation, we use our approach to explore large office environments. Office environments

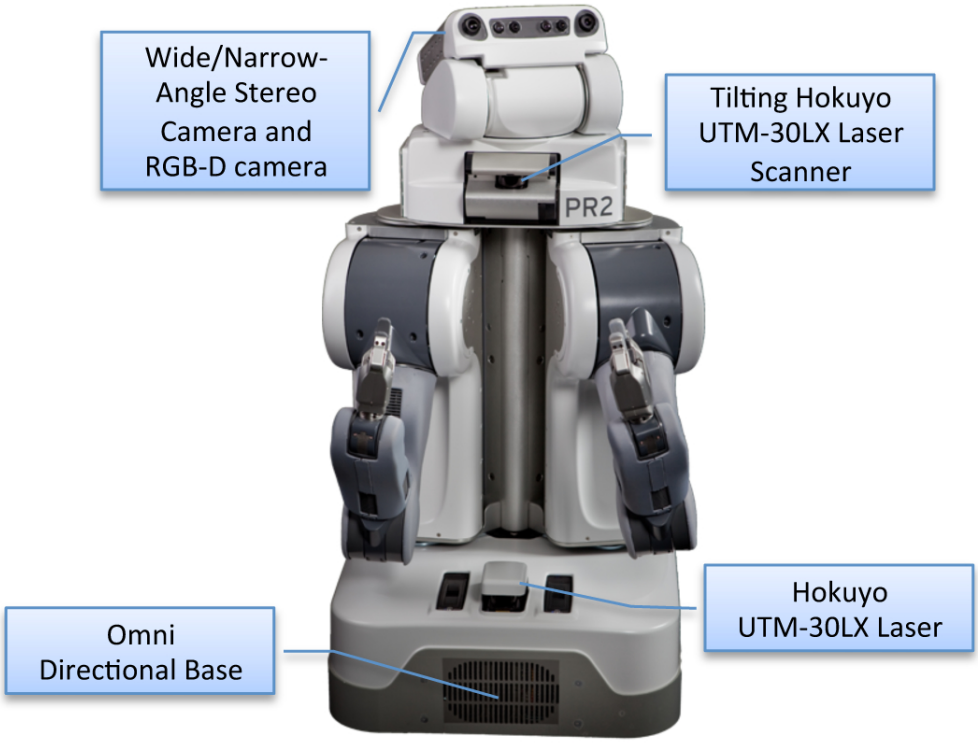


Figure 3.7: The experimental platform (PR2 Willow Garage). In this paper, we use the tilting Hokuyo UTM-30LX laser scanner, mounted above the shoulder and an RGBD sensor mounted on the head, for data acquisition. (Image, courtesy of Willow Garage)

are in general less cluttered and consist of large rooms and long hallways, with mostly free areas. We compare and evaluate our approach to different frontier-based exploration algorithms. Besides the size and the clutter of the two environments, they differ also in terms of viewing scale of the sensor. For detailed exploration, like the observation of objects on a table top, the sensor viewing scale is in general far bigger than the environment. This is different for large scale exploration usually, the dimensions of a room or a hallway exceed the sensor viewing scale.

For experimental evaluation we use the PR2 robot Fig. 3.7, a research and development platform, from Willow Garage, operated using the Robot Operating System (ROS) (Quigley et al., 2009). The PR2 robot has a mobile base, with a footprint of 668×668 mm, two manipulators as well as several high-end sensors (*e.g.* tilt laser scanner, stereo cameras, RGB-D sensor, IMU). The robot has a substantial size, which means the robot can not reach every position in space the next best view algorithm estimates. The consequence of this is almost always time consuming, since unreachability can only be determined in imminent vicinity of an unexpected object due to either unnecessary robot motions or

Figure 3.8: PR2 robot acquiring data from a table top. The robot is able to position itself anywhere around the table.

due to the need of recomputing a new next best view positions. We will explain in the following chapters how we deal with this issue and what effect this has on the information gain. The simulation experiments are performed using the robot simulator Gazebo. This simulator is able to replicate the PR2 robot with all its functionality in a physics simulated environment. In simulation we are utilizing the tilt laser scanner as well as a RGB-D camera for data acquisition.

In the following we evaluate our approach in simulation as well as on a real robotic system. First, we evaluate the detailed exploration of table top environment using a real robot. Second, we evaluate our approach in a large scale exploration environment of two office environments in a simulation environment.

3.3.1 Real Robot Experiments

To analyze the performance of our approach in small and cluttered environments, we simulate different cluttered table top scenes. The scenes, shown in Fig. 3.9, consists of different objects and configurations of the objects on the table, varying in different levels of complexity and clutter. The table is setup in a way that the robot has the ability to position itself anywhere around it, Fig. 3.8. To fully observe the table top and with that all the objects on it, we used the robot's mobile base to position the sensor around our region of interest. For the detailed data acquisition we are using a tilting Hokuyo UTM-30LX laser scanner, mounted on the upper torso of the PR2 robot, to produce the input point clouds. The laser scanner is more accurate and less noisy compared to a

RGB-D sensor, however it requires approximately 15 seconds to acquire a high resolution full 3D scan of the environment. Additionally, we restrict the lasers' field of view to 15 degrees to increase accuracy. This makes an environment measurement costly in terms of needed time, therefore the goal is to keep the number of needed scans minimal and reduce the unobserved space maximally for every new scan. Furthermore, the laser scanner is mounted in a way that the robot cannot see all the objects by looking down at the table. This makes the setup more general and harder to solve because not all objects are seen in the first scan.

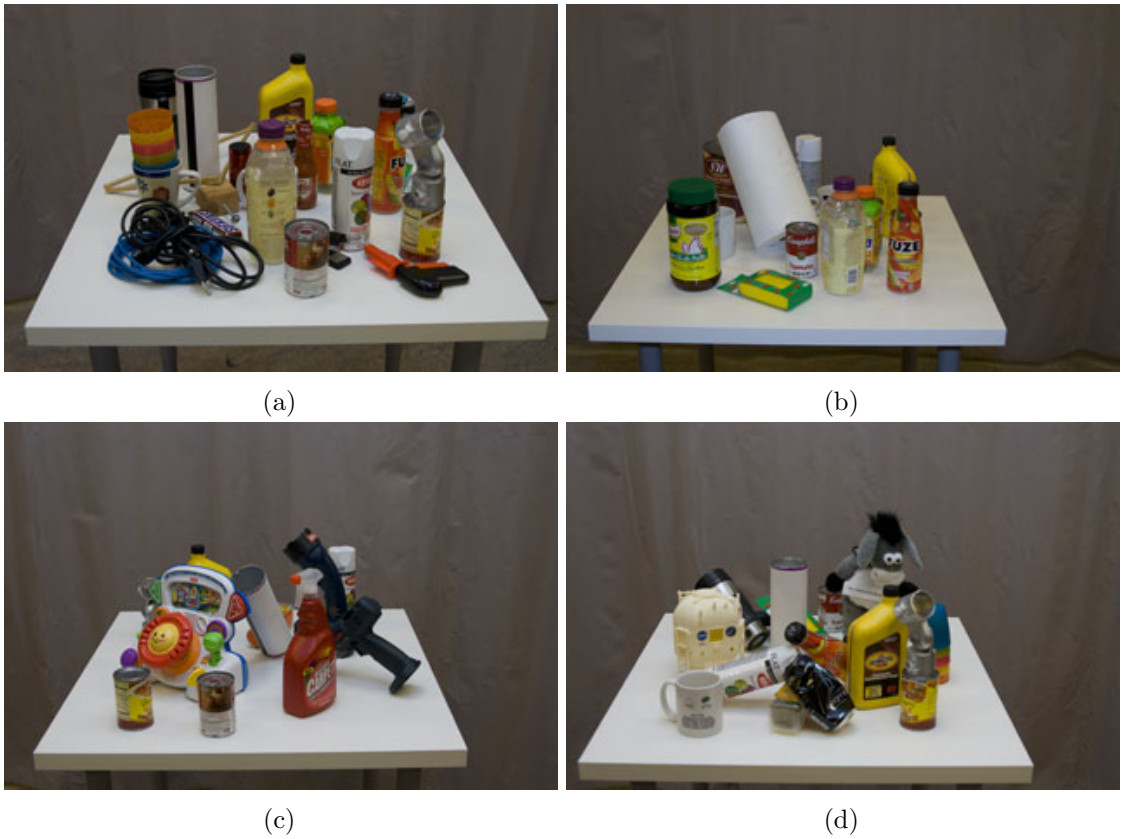
Since we are only interested in the objects on the table, we extract the objects from the table top as an additional pre-processing step. Given a new point cloud, the first step is to extract the table top and objects. The surface normals of all points in the point cloud are computed and are used to estimate all horizontal planar surfaces by using robust SAC (Sample Consensus) estimators. The approach is described in detail in (Rusu, Holzbach, Beetz, & Bradski, 2009) and implemented in the Point Cloud Library (PCL) (Rusu & Cousins, 2011). All points above this planar surface correspond to data points of objects. We only integrate measurement points belonging to objects into the occupancy grid.

The dimension $(w \times d \times h)$ of the occupancy grid influences the resolution of the final measurement acquisition result as well as the runtime of the exploration algorithm. Given the relative small table top with 1 by 1 meters, we choose the resolution of the occupancy grid as follows: $w = 129$, $d = 129$ and $h = 65$. The resolution of the occupancy grid influences the runtime of the algorithm greatly, since discretizing the space in smaller cells results in an increase of unobserved cells. More unobserved cells means more cells to traverse and process using the ray-traversal algorithm and with that an increases run-time. However, discretization size also influences the level of detail we observe the environment, in general we decided to have a high resolution for detailed observation while a low resolution is sufficient for observation of large space, for example. The resolution we picked for the detailed observation of the table-top results in a cell size smaller than 1 cm, which resulted in a very dense point cloud.

To execute the Markov Random Field preprocessing step to fill in missing measurements of the objects, the ICM algorithm requires the parameters η and β. They influence the convergence behavior and are set to $\beta = \eta = 1.0$ for a conservative convergence. The parameters influence the weight every relationship in a clique receives. Setting the parameters to $\beta = \eta = 1.0$ results in an equal weight. If we would set $\beta = 0$, we effectively remove the link between the paris of neighboring hidden variables. This would result in the global most probable solution is $s_{x,y,z} = o_{x,y,z}$.

Additionally the estimation of the NBV position depends on the parameter a, influencing how far we suspect to see into the unobserved regions. Best results were achieved with the parameter set to $a = 0.997$, which was empirically estimated.

In the detailed exploration experiments the NBV estimation depends on the number of possible sampling positions K and the angle θ of each s_k. We choose a finite set of viewing positions in such a way, that all of them lie on the perimeter of a circle around the region of interest. We have found that the discretization does not effect the result in



(a) (b)

(c) (d)

Figure 3.9: In the images (a)-(d) we show the experimental test scenarios for the detailed exploration. The table top scenes have different number of object simulating different level of clutter.

a significant way, since small pose changes of the sensor do not effect the field of view in a notable way. However discretization drastically shortens the computation time. For our experiments, we have set $K = 10$ and θ_i with $i = [1, 2, 3]$ with the angles [-15, 0, 15] resulting in 30 different viewing positions.

We compare our probabilistic framework with two other methods. The simple greedy approach assumes every unobserved cell is visible from a virtual scanning position, unless it is obstructed by an occupied cell. This predicted measurement z is then used to estimate and evaluate the information gain of a virtual scanning position $s_k \in \mathcal{S}$. This models the approach used in (Stachniss & Burgard, 2003). The second tested approach chooses the NBV position randomly after a new real measurement is taken by the sensor. To compare the three approaches we contrast the actual increase in information gain, which is equivalent to the decrease of unobserved cells in the occupancy grid. Additionally, we evaluate our approach in terms of prediction error of the information gain. A large prediction errors means overestimation of the information gain for a virtual viewing position. This has

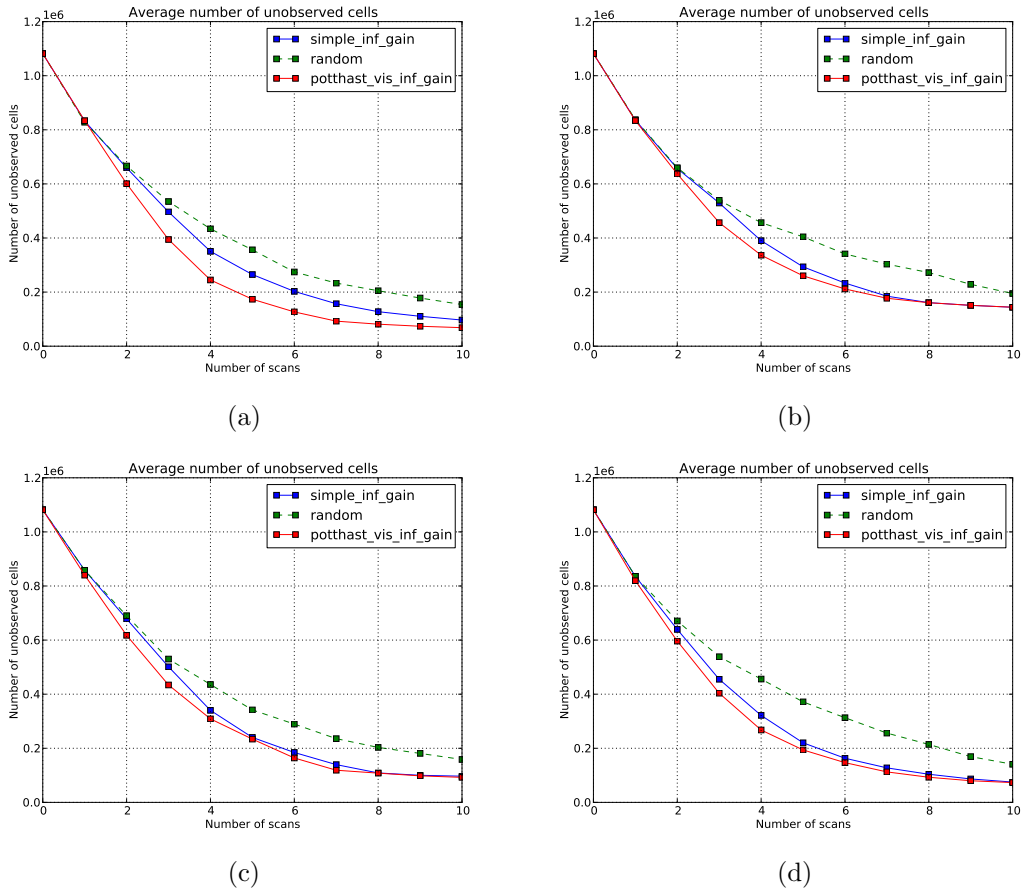


Figure 3.10: In the graphs (a)-(d) we compare the simple approach which is drawn as blue line with our proposed method drawn in red and a random approach in green. We can see that our method drawn in red decreases the number of unobserved cells faster, which is the effect of choosing NBV positions more optimal.

the affect that the chosen NBV positions is not necessarily a good one. Furthermore to give statical significance to our evaluation we conducted multiple runs for each scene and method. We conducted ten different runs for each scene and method, initialized from each of the sampling positions.

Evaluation

The overall performance shown in Fig. 3.10 shows the average decrease in unobserved cells over the 10 trials for each scan. We can see that our proposed approach converges faster and has therefore reduced the number of unobserved cells faster than the simple greedy approach. This also means that to observe a certain percentage of the unobserved space,

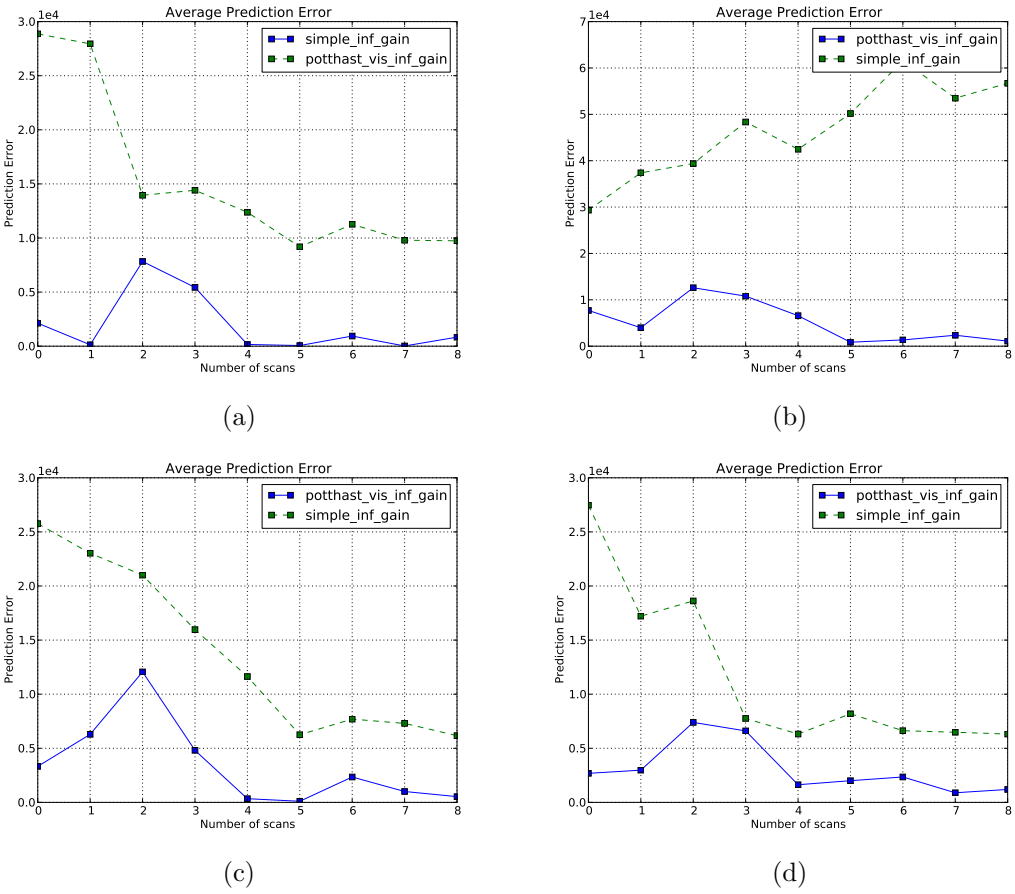


Figure 3.11: In the graphs (a)-(d) we show the absolute prediction error over the number of scans. The error is computed by taking the absolute error between the predicted information gain and the actual number of seen cells after a scan is taken. The green line in the graph represents the simple method while the blue line shows the prediction error of our method.

our approach can perform this in most cases much faster than the simple greedy method. As a consequence this means our method has chosen better next best scanning positions. An example of chosen positions and their reductions in unknown space can be seen in Fig. 3.12. Intuitively, the simple approach heavily over estimates the expected number of cells it can see from a virtual scan. As a result, it chooses next best scanning positions which decrease far less unobserved cells than expected due to objects, which have not yet been seen, blocking space which thought to be observable.

To chose the NBV position we have to predict the outcome of a virtual scan. In Fig. 3.11 we show that the absolute prediction error of our method is much lower than the prediction error of the simple method. The prediction error is calculated as the difference

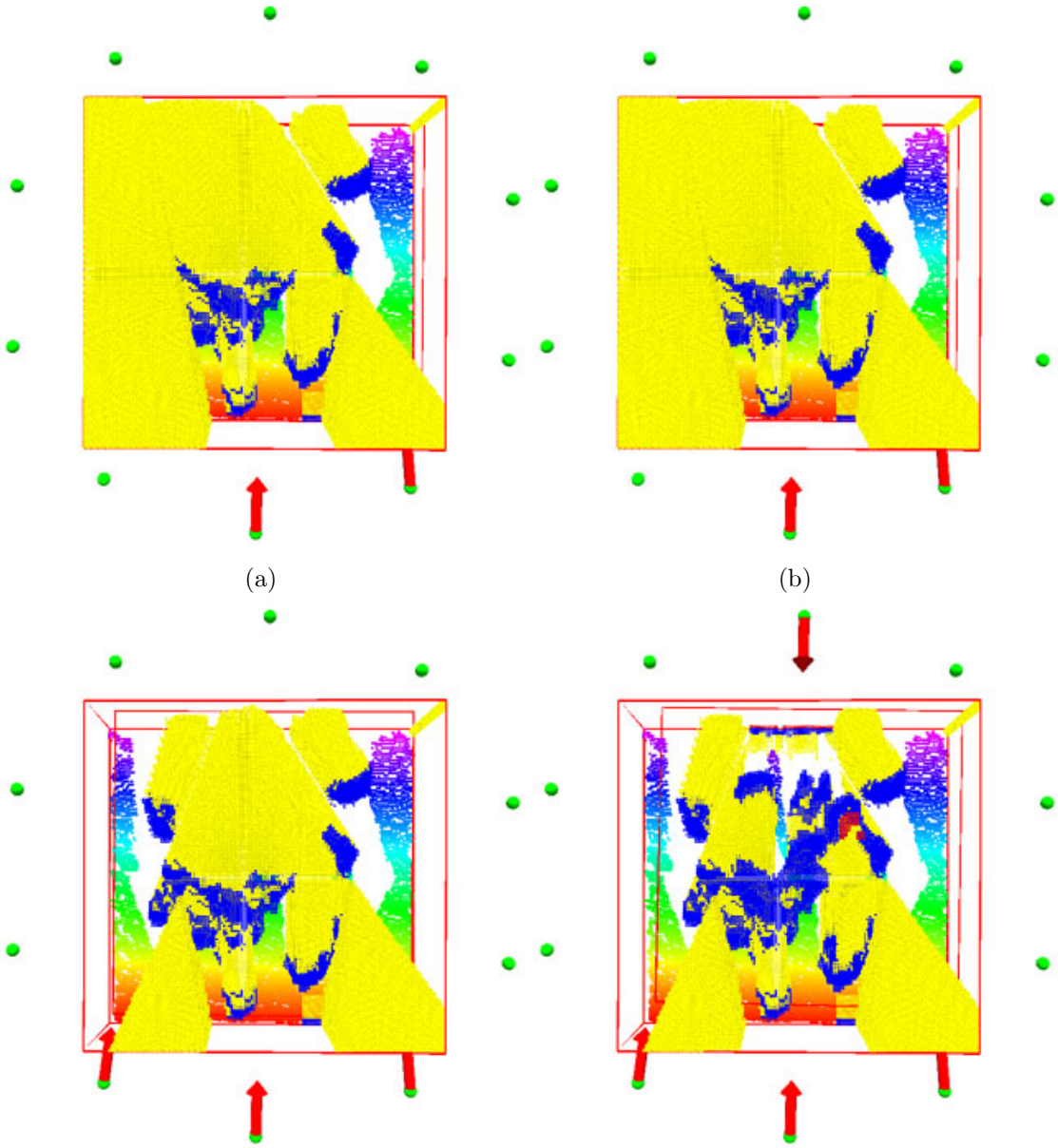
between the expected number of hitherto unobserved cells seen and the true number of hitherto unobserved cells seen in the new (real) scan. Note that due to the nature of the methods the two approaches computed two different sequences of sensor locations. Since this would result in an unequal comparison we ran the simple approach on the sequence estimated by our proposed method. Resulting in a comparison of the errors for the same scanning poses.

3.3.2 Simulated Experiments

In the second experiment we apply our approach to two different large scale office environments, consisting of hallways and several rooms. Both environments are less cluttered, compared to the table top scene, but substantially larger. Due to the size of the environment the main difference to the first experiment, besides the level of clutter, is the fact that the maximal distance a sensor can obtain a measurement in, is generally significantly smaller than in the table top environment. The office environment as well as the robot is simulated using the robots simulator Gazebo. We constructed two large scale office environments shown in Fig. 3.13 and Fig. 3.14. The first environment consists of two adjacent large rooms with a total size of 15 x 10 x 2 meters. The second environment is larger in size (20 x 25 x 2 meters) and consists of long hallways, small and large rooms. Furthermore, our approach is tested with different levels of clutter and varying starting locations of the robot. To explore the office environment we are using a simulated PR2 robot equipped with a simulated RGB-D camera (Microsoft Kinect). We limit the maximal sensor distance to 3.5 meter, which is approximately the real maximal distance of the sensor. Additionally, Gaussian noise is added to the measurement by the simulator. Although the RGB-D sensor is capable of continuous data acquisition, we constrain our exploration such that the robot has to be stationary to obtain measurements. To fully observe the environment the robot needs to iteratively estimate NBV positions, position itself and the sensor at the new location and acquire new measurements until the space is observed.

Due to the significant larger environment we initialize the occupancy grid with a higher resolution of $w = 257$, $d = 257$ and $h = 65$. The parameter for model optimization, especially for the ICM algorithm, remain the same as in the first experiment. The visibility parameter a of our approach is set to 0.9999, resulting in little visibility penalties when looking into unobserved regions. In general, office environments consists of long and clutter free hallways as well as open spaces in offices.

Rather than relying on pre-determined or sampled potential new viewing positions we estimate new positions on a frontier-region. Frontier-regions are boundaries between observed free space and unobserved space. After a new point cloud is integrated into the occupancy grid, we estimate all boundary cells and cluster them to frontier-regions. Each centroid of a frontier-region is a potential new viewing position and will be evaluated for its expected information gain. On every NBV positions we capture a full 360 degree sweep of the environment and integrate the measurements into the occupancy grid. Over the past decades many frontier-based approach were developed, the most prominent ones are



(a) (b) (c) (d)

Figure 3.12: The figure shows, from left to right, the reduction in unobserved cells after estimating the NBV position and acquiring new measurements. With each new observation more information about the environment is revealed, resulting i a reduction in unobserved (yellow) space.

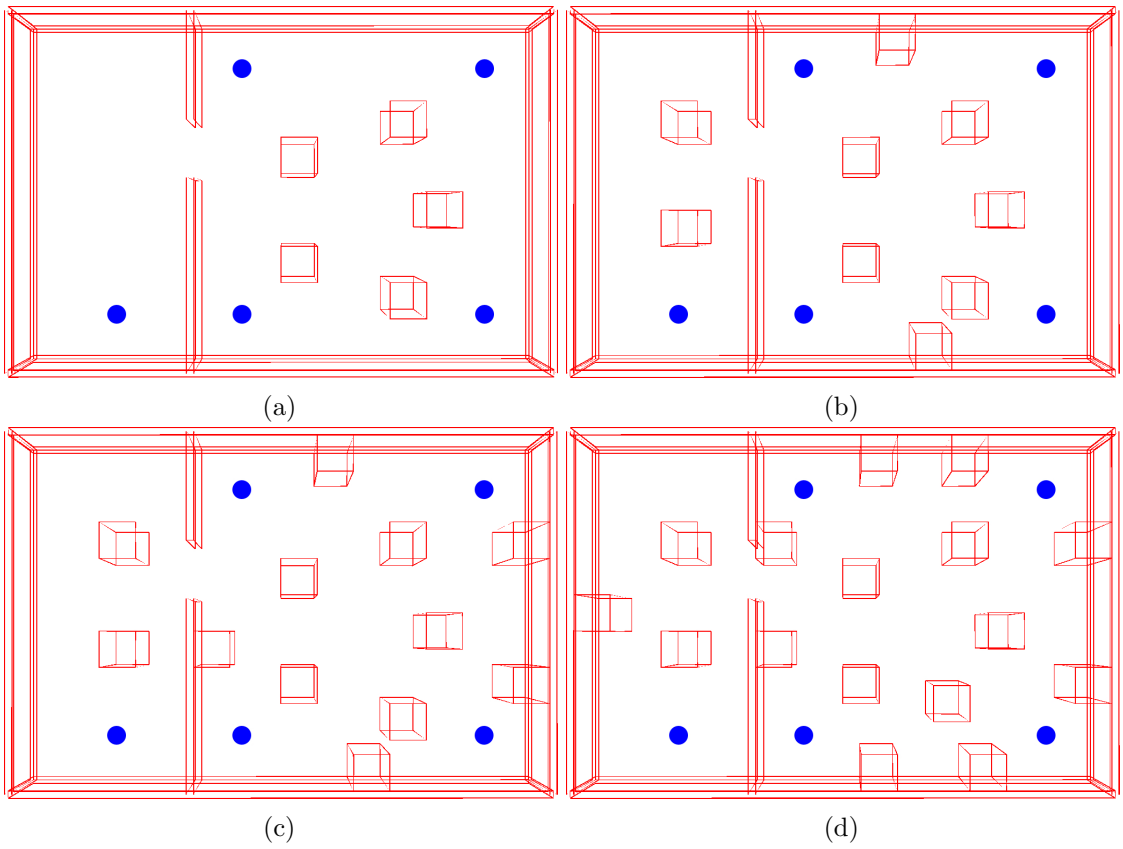


Figure 3.13: The figure shows a small office environment with different levels of clutter. For each environment we start the robot at five different initial starting locations, depicted as blue circles.

(Yamauchi, 1997) and (H. H. Gonzalez-Banos & Latombe, 2002) to which we compare our approach. Frontier approaches distinguish each other in the way NBV positions or frontiers are chosen and explored next.

The oldest but still widely used, because of it's simplicity, is *Yamauchi* (Yamauchi, 1997). The frontier centroid which is the closest to the robots current position is chosen as the NBV position. This strategy results in extremely short travel time after new measurements are acquired, however in general the number of necessary view points is of a factor of two larger compared to more sophisticated approaches. *Latombe* (H. H. Gonzalez-Banos & Latombe, 2002) developed a popular approach combining information theory and travel cost. For every frontier region the information gain is computed, with the measurement approximated by assuming infinite visibility, and penalized by the distance the robot would have to travel to get to that NBV position. The position with the highest reward is chosen as the NBV position and with that the next exploration position.

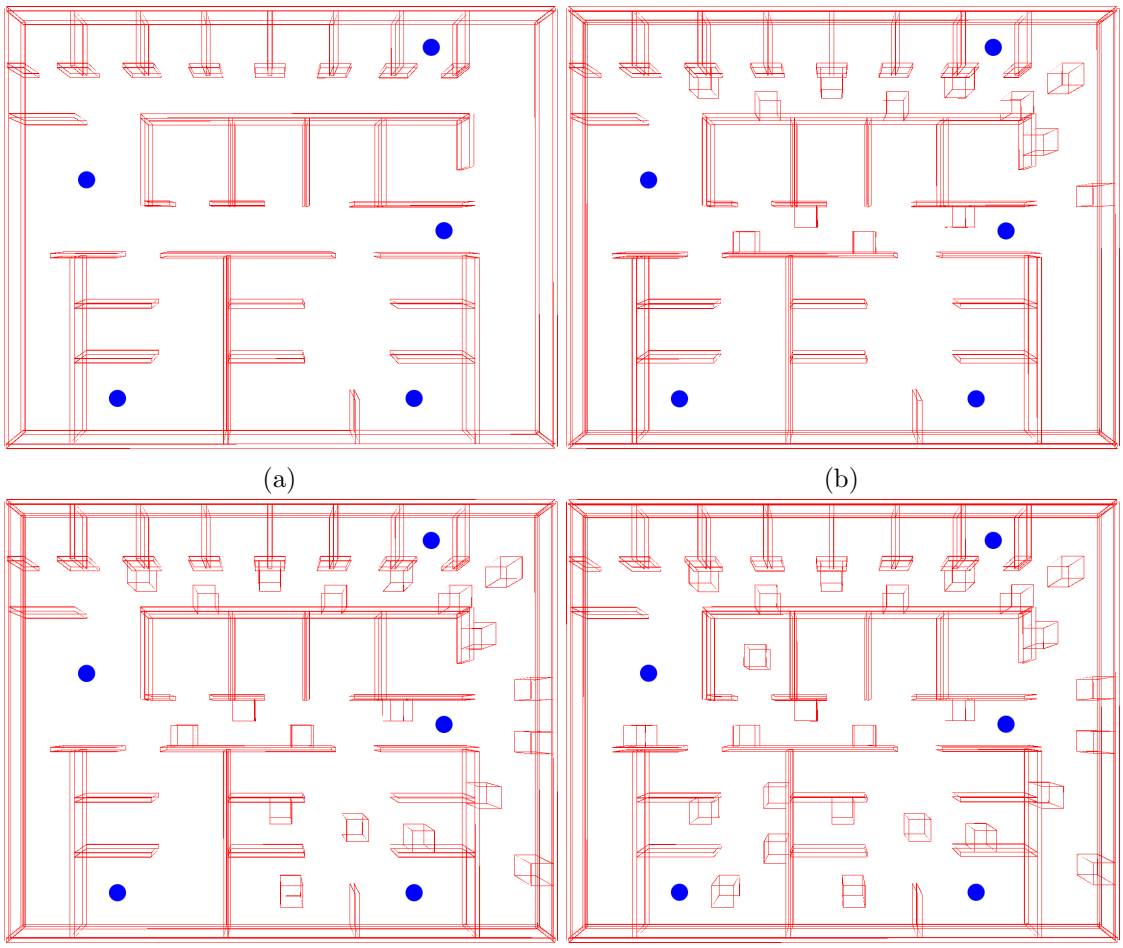


Figure 3.14: The figure shows a large office environment, with different levels of clutter. For each environment we start the robot at five different initial starting locations, depicted as blue circles.

The robot navigation algorithm tries to navigate the robot to the chosen goal point. However, not all NBV positions are reachable by the robot, since the estimated centroids of frontiers could lie within or very close to objects. In this case the navigation algorithm tries to get the robot as close as possible to the goal point, but eventually has to aboard and proceeds as if the goal position is reached.

Furthermore, we compare our approach to the simple greedy approximation of the information gain as described in the previous experiment. The frontier centroid with the highest information gain is chosen to be the NBV position. Finally, we compare our approach to a random selection of the frontier centroids.

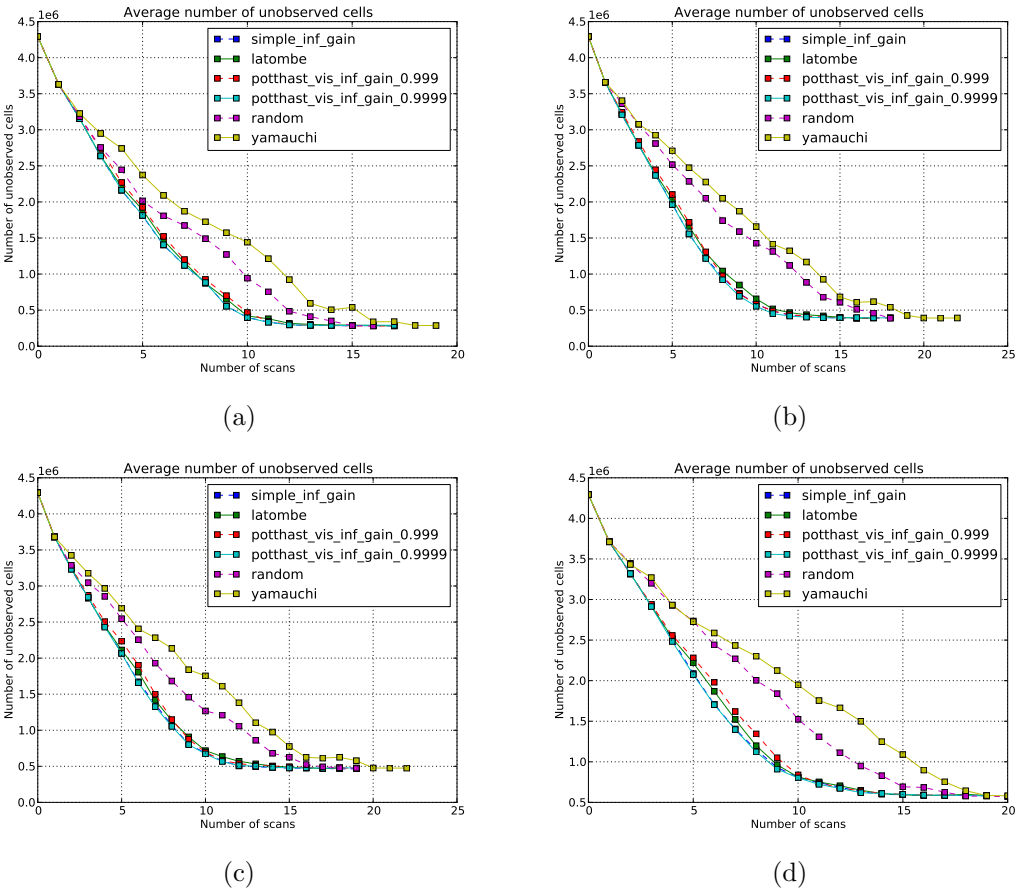


Figure 3.15: The figure shows the average performance for the small office environment Fig. 3.13 of our approach compared to the simple greedy, Latombe, Yamauchi and random. The observability parameter for our approach is set to $a = 0.999$ and $a = 0.9999$. We can see that our approach is on par with all the other approaches.

Evaluation

The performance for the different approaches in a large scale exploration task can be seen in Fig. 3.15 for the small office environment and Fig. 3.16 for the large office environment. As we can see, by setting the observation parameter close to 1.0 we converge to a similar behavior as the simple greedy information gain approach. Intuitively, it makes sense that due to the sparseness of the environment, the best exploration performance in terms of reducing the number of unobserved cells as quickly as possible, is achieved by assuming almost infinite visibility. In both office environments this seems to be the best strategy, and our approach has the flexibility of adapting this rather greedy strategy. For both office environments experiments have shown that the performance for different values of the

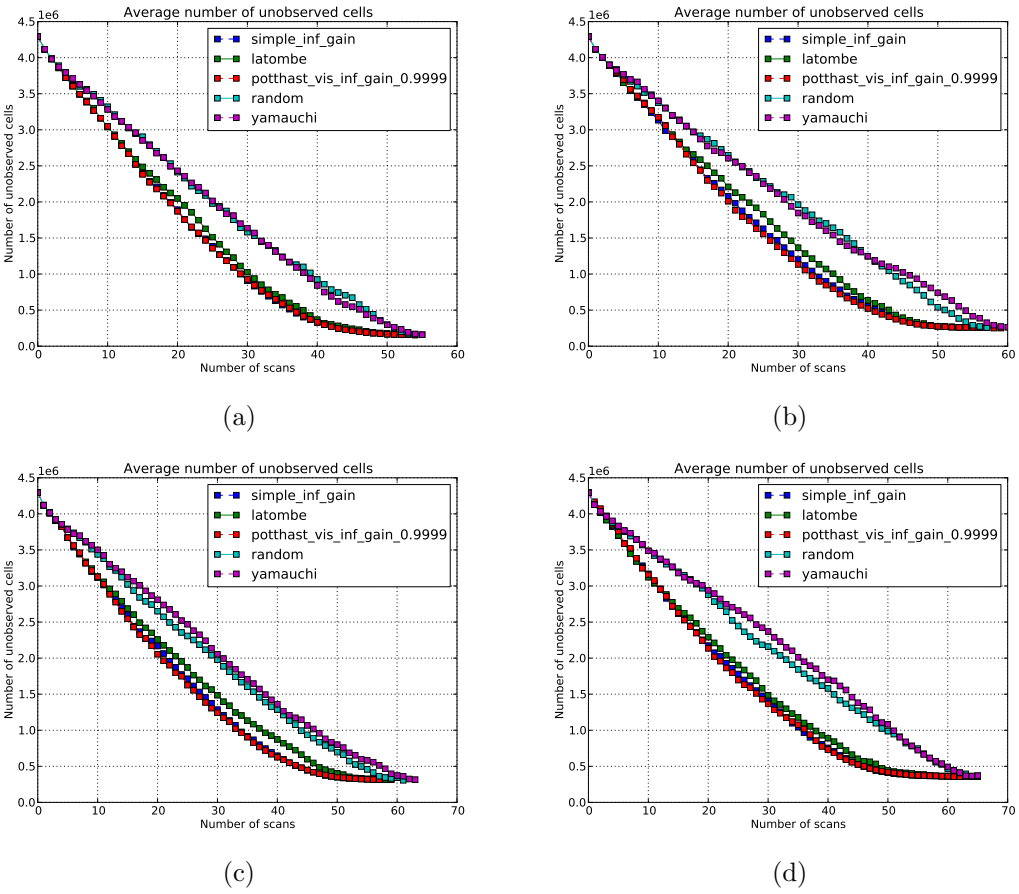


Figure 3.16: The figure shows the average performance for the large office environment Fig. 3.14. Our approach is at least as good as the other tested approaches. The observability parameter is set to $a = 0.9999$.

observation parameter does not degrade substantially. This is probably due to the fact that both environments are fairly large and sparse in obstacles. Hidden unobserved obstacles are rare and positions with large unobserved frontiers yield comparable performance.

3.3.3 Practical Issues

The next best view algorithm we have presented in this paper is evaluated on a robotic system, more specifically on the PR2 robot. In general, the proposed next best view estimation algorithm is independent to a specific robot or any underlining system, however we use it to acquire data and more importantly get an estimated of the sensor positions in a word coordinate frame. In the following subsections we will give some insight into the practical issues when using our specific evaluation system.

Robot Limitations

Depending on the system used to move the camera not all positions in space are reachable by the system. Our robot in particular is fairly large and can only operate on a 2D ground plane. Practically, this means potentially next best view positions can only be sampled from a subset of all possible viewpoints. This subset includes only the positions which lie within the robots configuration space. In contrast to our robot, a robot operating in 6D like an areal vehicle has a much larger operation space. However, even if we only sample in the robots configuration space, it is still possible that the positions sampled are not reachable. This happens if a potential viewing position lies within unobserved space and is obstructed by a previous unseen object. For the tabletop scene we assume there are no obstacles surrounding the table which could prevent the robot from reaching a sampled potential new viewing position. In the large environment exploration task this can however happen, since the algorithm we use to estimate potential new viewing positions can return positions which lie within unobserved space. If such a position can not be reached a fallback plan has to be used. In our case we attempt to get as close as possible to the position, treating this as the new viewing position and acquire a new environment scan. However, there are many other operations that can be performed, e.g. choosing a different positions from the set of potential new viewing positions. Problematic with our fallback plan is that the estimated expected information gain, estimated by our proposed algorithm, for the fallback position is different than for the original position. This means taking a scan on the fallback position has a worse information gain than estimated on the original planned position or even as on a different position in the set of potential next best viewing positions. Unfortunately, we only know this once we have actually taken a new observation and integrated in our world representation. Moving to a different positions once the robot is not able to reach the position is a viable option, but it is not clear that this has more benefits. First, the robot has a lot more cost to actually reach a new positions, it first has to drive to the original, if not reachable, it has to drive to another one. Second, we do not know if this position is in fact better than the one close to the unreachable first one, since without taking an observation at the unreachable first position we have no information of its expected information gain.

Registration Issues

Registration of sensor data over time is still an unsolved problem and a very active research field. In the real world experiment we have a tailored solution which works reasonable well and produces only small errors in registration. In the simulation environment, we do not assume perfect position knowledge of the robot, but rather apply gaussian noise to every position estimate. Registration errors accumulating over time will have an effect on the final representation of the world but also on the estimation of the expected information gain and with that on the choice of the next best view position.

Runtime

The runtime of the algorithm highly depends on the number of sampled potential new viewing positions and the total number of cells that need to be evaluated. The runtime of estimating the expected information gain of one potential next best view positions therefore highly depends on the discretization of the environment. This varies from a second to a couple of seconds, due to costly ray-traversal operations. Currently we have a fast implementation, however there are GPU implementation which would improve the speed tremendously. Each additional potential next best view, which needs to be evaluated for its expected information gain, increases the total runtime by the evaluation time. Typically, the complete evaluation for the best next best view points takes about 30 seconds. A future goal would be to push this evaluation to under 10 seconds.

3.4 Discussion

The results of the experiments show the usefulnesses of our approach in terms of detailed and large scale exploration. Comparing the results in the two different scenarios, one can see that reasoning over the observability of the cells is most useful in an extreme cluttered environment and when the sensor viewing scale is larger than the environment. In the detailed exploration setting our approach outperforms all other tested methods. On average, our proposed framework reduces the number of unobserved cells faster than the simple greedy method. In the large scale experiment the performance of our algorithm is at least as good as the simple greedy approach and is on par or better than the other tested approaches. We occasionally see a slightly faster reduction of unobserved cells using our approach, however in all of the cases it was insignificant.

When comparing our method to different approaches found in the literature the real advantage comes with that fact that we are are able to mimic several different exploration strategies within one approach. Changing the value of the observability parameter a and with that the observability assumption of an unobserved cell also results in different exploration behaviors. This allows us to adapt our exploration strategy, depending on the task, within the same algorithm. With that our approach is more general to different exploration scenarios and needs. When changing the observability parameter to a value close to 1.0 we achieve no penalty for unobserved cells, resulting in the assumption of infinite visibility. In comparison when we set the parameter closer to 0.9 we achieve a behavior which mimics the behavior of always exploring the biggest frontier. Values in between 0.9 and 1.0 result in more or less penalization for unobserved cells.

A limitation of our approach as presented here is that the observability parameter a needs to be set manually, depending on what kind of scene needs to be explored. However, setting the parameter is very intuitive, for very cluttered scenes a value of $a \simeq 0.997$ is a good starting point. For large scale scenes with relatively few clutter, a value of $a \simeq 0.9999$ will most likely work well.

3.5 Conclusion

We have presented a flexible probabilistic framework for Next Best View estimation. We have shown that our approach can mimic several different explorations strategies. Thus the proposed framework is a more general approach to NBV estimation. This generality has been validated in several experiments conducted in simulation as well as on a real robotics platform. In both experimental settings, the detailed exploration and large scale office exploration, we can show that our approach is on par or better than other popular approaches like the simple greedy method.

Chapter 4

Active Multi-View Object Recognition and Online Feature Selection

Despite decades of research, object recognition from visual percepts is still an unsolved and challenging problem. In robotics especially, where many tasks build on, or directly involve, the recognition of objects, the need has grown for creating reliable and fast classification systems. Conventional approaches for object recognition are tour de force, exclusively relying in large part on complex statistical models for classification and heavy engineering of computationally-intensive features (Bo et al., 2011; Lowe, 2004). All too often, systems are assumed to be unlimited and their processing capabilities are neglected. This stands in stark contrast to many of today's mobile robots, which often have limited power and need to be economical with resources for computing and reasoning—being able to move and execute given tasks is their first priority!

Active object recognition(Atanasov, Sankaran, Ny, Pappas, & Daniilidis, 2014; Denzler & Brown, 2002) is an appealing paradigm to overcome the challenges in recognition of objects. Here, a mobile robot configures and positions its sensors to inspect a target object and recognize it as a previously learnt object. The advantage of active object recognition lies in the ability of reacting to unforeseen objects or scenes by exploratory actions. This allows to seek actively for the information that is the most useful in performing inference under uncertainty.

Adaptive action selection offers the flexibility that is required to potentially re-use the statistical modeling power of conventional approaches even under the severe (computing) constraints present on mobile robots. The general idea is to control the sensors such that only data with high information content is collected. By running inference recursively based on previously acquired information (which represents the robot's current knowledge, i.e., its belief state about the identity of the object), the collection of data can be *adapted dynamically*. Therefore, computation time can potentially be reduced, because we only process useful information and have the ability to improve our belief with each additional action.

In previous work, there are two common actions that have been explored in order to increase the recognition accuracy: first, feature selection (Verleysen & Rossi, 2009),

which computes a subset of informative features; second, viewpoint selection or view planning (Potthast & Sukhatme, 2014; Atanasov et al., 2014), which controls where and how many additional observations to take. However, previous approaches have not looked at the potential of reducing computation time due to feature selection. Moreover, existing work does not exploit the full potential of combining these two strategies and applying feature and viewpoint selection in concert.

The main idea of our work hinges on the following intuition: *Is it possible to reduce overall costs for recognition by relying on feature subsets gathered from multiple views*, in lieu of the full feature set, potentially taken from a single view, using a complex statistical model for classification? More concretely, in this paper, we present an information-theoretic multi-view object recognition framework that unifies the two strategies of *feature selection* and *viewpoint selection*. We aim at exploiting local information by selecting additional features at the current viewpoint and further exploring the object by selecting new viewpoints around it. The described framework optimizes the viewpoint selection by reducing ambiguities and selecting features dynamically, such that only useful and informative features are extracted and incorporated in the recognition process. Unlike standard feature selection, the two strategies are symbiotic in our algorithm. Feature selection is an iterative process, thus at any given time, the selected feature depends on the trajectory of the past viewpoints while simultaneously affecting the future viewpoint(s).

We perform extensive evaluations on a large RGB-D camera dataset, which is composed of single object scenes. To evaluate our framework, we conduct experiments with two different feature sets, one set consisting of *simple features* designed by us, and one set of more complex *kernel descriptors* proposed by (Bo et al., 2011). The use of the simple features shows that, in combination with feature selection and view planning, our framework can compete with state-of-the-art single-view classification systems. In contrast, using the kernel descriptors shows that our framework is capable of dealing with more advanced features as well.

We further show the advantages of combining intelligent view planning and online feature selection. Online feature selection reduces the computation time fivefold for the simple features and 2.5-fold for the kernel features at runtime. Furthermore, in combination with view planning, we are able to increase the recognition accuracy significantly by resolving object ambiguities. In the case of the simple features, we achieve an increase in accuracies by 8–15%, while in case of the kernel descriptors the increase is about 8%. Overall, our results show pose estimates within $\pm 45^\circ$ of the ground truth and object recognition accuracies of over 90% when using the simple features in rather challenging object recognition experiments. Using the kernel descriptors, we improve the object recognition accuracy to over 98%. As proof-of-concept, we eventually implemented and tested the proposed approach on a custom-built quadcopter robot. The quadcopter is equipped with an RGB-D camera and collects multiple views by flying around a target object, which allows to recognize the object.

Additionally, we show how our active multi-view recognition system can be used for *object change detection*. Given a prior map of the environment, the task is to determine

whether objects in the environment have changed. This can either mean that the object has changed completely (i.e., it belongs to a different object class), or the object has changed its orientation only (i.e., it belongs to the same class but its rotation differs in yaw). Our results show that on average our framework can detect such changes in the environment with an accuracy of larger than 90%.

Prior works on active perception go back as far as the seminal paper (Bajcsy, 1988). "Active" can either refer to adjustments of the sensor parameters themselves, e.g., adaptively changing the focal length of a camera, as in (Bajcsy, 1988; Denzler & Brown, 2002), or mean that the entire sensor moves together with the underlying platform. The latter is particularly common in robotics, where robots plan new observation locations and viewing directions actively (Laporte & Arbel, 2006; Hollinger, Mitra, & Sukhatme, 2011; Atanasov et al., 2014). This is known as robotic view planning. (Potthast & Sukhatme, 2014) presents a comparison of several representative approaches to view planning from literature. View planning can be formulated as a purely geometric problem (K. Tarabanis, Tsai, & Kaul, 1996; H. Gonzalez-Banos & Latombe, 1998) but as well relates to robotic exploration, where information-theoretic strategies have become state-of-the-art solutions (Stachniss, Grisetti, & Burgard, 2005; Krainin et al., 2011). The selection of optimal viewpoints for a robot can be useful in various applications, such as for the inspection of underwater structures (Hollinger, Englot, Hover, Mitra, & Sukhatme, 2013), for object search in indoor environments (Ekvall, Jensfelt, & Kragic, 2006; Finman, Whelan, Kaess, & Leonard, 2013) or detecting objects on a table top (Atanasov et al., 2014). Moreover, methods for view planning and active object recognition provide the basis for operations like object sorting (Gupta, Muller, & Sukhatme, 2015) and manipulation (Krainin et al., 2011; Holz et al., 2014), acquisition of object representations (Welke, Issac, Schiebener, Asfour, & Dillmann, 2010) or change detection (we will show in the end of Section 3.3 how our object recognition framework naturally extends to perform object change detection).

The common goal of object recognition approaches is to recognize objects and estimate their poses (Denzler & Brown, 2002; Laporte & Arbel, 2006; Ekvall et al., 2006; Collet & Srinivasa, 2010; Lai, Bo, Ren, & Fox, 2011). (Denzler & Brown, 2002) presents an active object recognition framework that, similar to ours, relies on a sequential decision process and mutual information as information-theoretic measure for the selection of actions. (Laporte & Arbel, 2006) selects multiple viewpoints by using the Jeffrey's divergence instead of maximizing the mutual information. (Ekvall et al., 2006) combines active object recognition with Simultaneous Localization and Mapping (SLAM) techniques to estimate object positions and link semantic with spatial information. (Collet & Srinivasa, 2010) extends a single-view to a multi-view object recognition method for increased efficiency by applying multi-step optimization and extracting SIFT features from camera images. (Lai et al., 2011) performs object recognition and object detection by using different classifiers and several shape and visual features.

To cope with objects that appear ambiguous in single views, (Borotschnig, Paletta, Prantl, & Pinz, 1998; Paletta & Pinz, 2000; Brown & Lowe, 2005; Collet & Srinivasa, 2010) propose multi-view object recognition frameworks that use cameras and image databases,

whereas (Atanasov et al., 2014; Finman et al., 2013; Luber, Spinello, & Arras, 2011; Lai et al., 2011) recognize objects by extracting features from laser scanners and RGB-D depth cameras. (Luber et al., 2011) applies the online boosting approach from (Grabner & Bischof, 2006) to point clouds in the robotic domain for feature selection. Feature selection can support the object recognition task by reducing the computation time and confusions through suboptimal feature sets (Guyon & Elisseeff, 2003; Verleysen & Rossi, 2009). A standard alternative to boosting is the selection of features by mutual information (Peng, Long, & Ding, 2005; Verleysen & Rossi, 2009). (Peng et al., 2005) tries to select the most relevant features by maximizing the dependency of a feature on the target object (i.e., maximizing the mutual information), while minimizing the redundancy between the features. (Ali & Marton, 2014) evaluates the feature selection methods from (Peng et al., 2005) when applied to the RGB-D dataset of (Lai et al., 2011). Ensemble methods (Dietterich, 2000) offer further alternatives of how to combine features; (Schimbinschi, Schomaker, & Wiering, 2015) employs ensemble methods for 3D face recognition and (Marton, Seidel, Balint-Benczedi, & Beetz, 2013) compares the concatenation of all available features to ensembles with stacking. (Zhou, Comaniciu, & Krishnan, 2003) proposes a unifying view on active recognition and feature selection, however, focuses on feature interdependencies and the selection of features given a concrete context, e.g., a specific target object to detect. Their method, as well as the other feature selection methods for object recognition mentioned above, do not explicitly consider viewpoint selection, whereas our object recognition framework applies both feature and viewpoint selection in concert.

Although many approaches for active view planning as well as online feature selection exist, to the best of our knowledge, the advantages of both approaches have never been leveraged in combination. This paper builds on a prior conference paper (Potthast et al., 2015). The journal version provides a more detailed problem formulation, demonstrates broader applicability to more complex feature sets and object change detection, and includes more extensive evaluation of the unified feature and viewpoint selection framework.

The remainder of the article is organized as follows. The problem formulation and proposed solution of our active multi-view object recognition approach is detailed in Section 4.1. Section 4.2 introduces the unified selection of actions, which is applied to both feature selection in Section 4.3 and viewpoint selection in Section 4.4. The object recognition experiments and results are discussed in Section 4.5, and a conclusion is provided in Section 4.6.

4.1 Problem Formulation

We look at solving the following problem. Given a static scene and an object in question, we want to recognize the object and, at the same time, estimate its pose relative to the coordinate system of the observing sensor, while keeping the computation cost to a minimum. The recognition process matches an observation of the target object with the objects stored in a database and computes the belief states of all the objects. Due to

noise, ambiguity and occlusions the recognition process does not always lead to certain matchings, which in turn results in multiple likely object candidates. To this end, our process seeks to select a minimal sequence of actions which reduces the computation time but simultaneously maximizes the acquired information about the target object. In other words, the actions are chosen in such a way as to reduce the uncertainty in the current belief state estimates for the object candidates.

4.1.1 System Overview

The general system diagram of our active recognition framework is presented in Fig. 4.1. It consists of the four modules: *object observation*, *feature selection*, *recognition state estimation*, and *viewpoint selection*.

In each time step t, we select a next action from the set of feasible actions, which are: 1) stopping the recognition process, 2) staying at the current viewpoint and computing a next feature f_{t+1} (feature selection), or 3) moving the sensor to a new viewpoint by applying the motion action a_t (viewpoint selection). Concretely, the action a_t defines the relative motion of the sensor to move from the current observation position to the new observation position. We only allow sensor motions to a restricted set of next observation positions. The discretization of the viewpoints reduces the complexity in computation when compared to a continuous representation of viewpoints. When computing a new feature f_{t+1} at the current viewpoint, we set $a_t = \varnothing$, defining a *zero motion action*.

During object observation, the sensor captures the raw data, such as color images and 3D point clouds, which is used to compute the features. The sensor measurements at a new viewpoint are parametrized by the feature vector $\vec{f} = [f^1, \ldots, f^N]$, which includes N different feature types. Given that the n^{th} feature in $\vec{f}$ has been selected at time t, we extract feature $f_t = f^n$ from the measurement data. The observation uncertainty is captured in our observation model, defined as the conditional probability density function $p(f_t|o_t)$, with recognition state $o_t = \{c_t, \theta_t\}$. Here, the feature f_t is conditioned on the object class c_t and orientation angle θ_t of the observed object. In the recognition state estimation, we use a *Bayesian framework* to infer about the current c_t and θ_t from the measurements. The parameters are assumed to be obtained from a physical object model and to be trained offline.

At each observation position, we start off with extracting one first feature from our total feature set specified by vector $\vec{f}$. We then have the choice of local exploitation, i.e., extracting another feature, or global exploration, i.e., moving the sensor to a new observation position. The premise is to extract as much information as possible at the current viewpoint since a motion action is inherently more costly. Only after having exploited all the features that yield useful information, we want to move to a new location. However, as moving the sensor physically is costly, finally, we move to the next viewpoint only if the gain of information (in expectation) significantly reduces the uncertainty in the object belief states. Put in other words, a next motion action is applied only if the recognition accuracy is expected to improve by a significant amount.

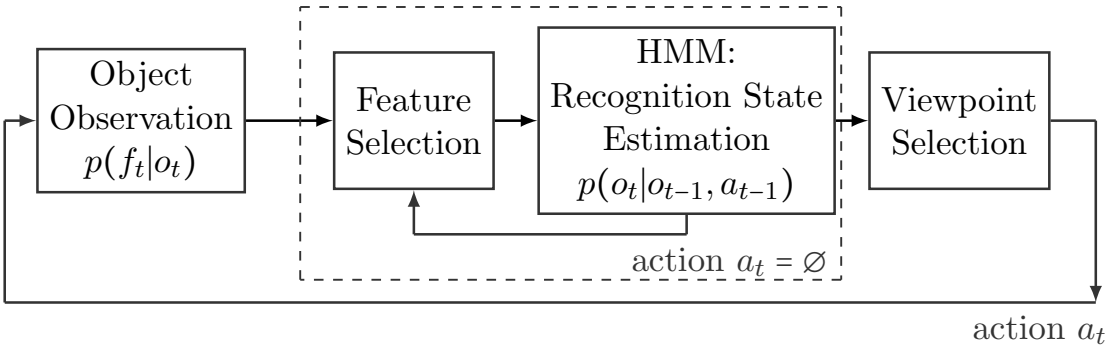


Figure 4.1: System overview of the active multi-view object recognition framework. After each observation, the HMM updates the recognition state, upon which a new feature (inner loop) or a new viewpoint (outer loop) is selected. The feature type that is selected first at any new location is pre-defined. The iterative processes terminate when the stopping criteria are met.

Before we extract and add a new feature, we compute the utility of all remaining features in our arsenal and select the one with the highest utility. We only allow to add one feature at a time, and once a feature is added it cannot be removed from the set anymore. This way, integrating a new feature simply becomes a sequential update step without the need for re-computing the posterior of the entire feature set. We continue until either a stopping criterion has been reached or all N features are observed at a given viewpoint, forcing us to either terminate or move to a new viewpoint. The overall idea is that, at each viewpoint, we have only extracted useful features and left out features that yield no additional information. Moreover, not extracting a feature saves time, which helps to reduce the total time needed to recognize the object.

4.1.2 Sequential Bayesian Recognition

With every new observation of a feature f_{t+1}, we want to efficiently integrate the new observation with results from prior observations, and thus update our current belief about the object class and orientation. We formulate the recognition and pose estimation over the object classes and object poses as a Bayesian network in form of a *Hidden Markov Model* (HMM). A Bayesian formulation has distinct advantages, with the main advantage being that Bayesian methods model all sources of uncertainty in form of random variables. Furthermore, HMMs have efficient formulations to sequentially update the poste rior distribution, making it ideal for integrating new observations, such as in our case in the form of a new feature f_{t+1}. Compared to a naive Bayes update, which is often used in existing work (Denzler & Brown, 2002; Paletta & Pinz, 2000; Laporte & Arbel, 2006; Borotschnig et al., 1998), the HMM formulation additionally gives us the possibility to explicitly formulate the *state transitions*.

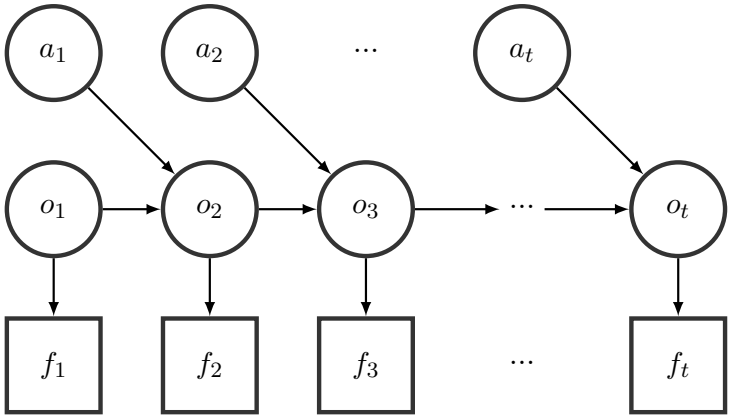


Figure 4.2: HMM for sequential Bayesian recognition. Given a sequence of observed features $f_{1:t}$ and executed actions $a_{1:t}$, we reason about the hidden recognition state of an object at time t, $o_t = \{c_t, \theta_t\}$, including its object class $c_t = c^k$ and object pose $\theta_t = \theta^q$.

The joint probability distribution of the HMM, as shown in Fig. 4.2, is defined as

$$\begin{aligned} P(o_{1:T}, f_{1:T}, a_{1:T}) &= P(o_{1:T})\, P(a_{1:T}|o_{1:T})\, P(f_{1:T}|o_{1:T}, a_{1:T}) \\ &= \left[P(o_1) \prod_{t=2}^{T} P(o_t|o_{t-1}, a_{t-1})\right]\left[\prod_{t=1}^{T} P(f_t|o_t)\right], \end{aligned} \tag{4.1}$$

and the corresponding log probability, $\log(P(o_{1:T}, f_{1:T}, a_{1:T}))$, equals

$$\log(P(o_1)) + \sum_{t=2}^{T} \log(P(o_t|o_{t-1}, a_{t-1})) + \sum_{t=1}^{T} \log(P(f_t|o_t))\,. \tag{4.2}$$

The observation sequence is $S = \{o_1, \ldots, o_t, a_1, \ldots, a_t\}$, the hidden state of our HMM is defined as recognition state o_t with $o_{1:T} = \{o_1, \ldots, o_t, \ldots, o_T\}$, the observations are given as observed features f_t with $f_{1:T} = \{f_1, \ldots, f_t, \ldots, f_T\}$ and the motion actions between viewpoints are given as a_t with $a_{1:T} = \{a_1, \ldots, a_t,$
$\ldots, a_T\}$. The state of the hidden process satisfies the Markov property, that is, given the value of o_{t-1}, the current state o_t is independent of all the states prior to $t-1$.

The *observation model* $p(f_t|o_t)$ represents the likelihood that feature f_t is generated by recognition state o_t. The transition probability is defined as $P(o_t|o_{t-1}, a_{t-1})$, which means that the transition to the next recognition state is dependent on the previous recognition state (i.e., the previous object class and orientation) as well as the latest motion action that has been applied. The posterior marginals $P(o_t|f_{1:T}, a_{1:T})$ of all hidden state variables, given a sequence of observations, can be efficiently computed using the *forward-backward*

algorithm. The algorithm computes a smoothed estimate given all evidence in two passes, with the first pass going forward in time and the second pass going backwards in time:

$$\begin{aligned} P(o_t|f_{1:T}, a_{1:T}) &= P(o_t|f_{1:t}, f_{t+1:T}, a_{1:t}, a_{t+1:T}) \\ &\propto P(f_{t+1:T}, a_{t+1:T}|o_t)\ P(o_t|f_{1:t}, a_{1:t})\,. \end{aligned} \tag{4.3}$$

In the first pass, we compute the forward probabilities, which are the probabilities for all $t \in \{1, \ldots, T\}$ ending up in any particular state given the first t observations in the sequence $P(o_t|f_{1:t}, a_{1:t})$. In the second step, we compute the probabilities $P(f_{t+1:T}, a_{t+1:T}|o_t)$ of making the remaining observations given any starting point t.

The recursion of the forward-backward algorithm can be broken up into a forward and backward part. The forward recursion is defined as:

$$\begin{aligned} \alpha(o_T) &\equiv P(f_{1:T}, o_{1:T}, a_{1:T})\,, \text{ and} \\ \alpha(o_T) &= P(f_T|o_T) \sum_{t=2}^{T} \alpha(o_{t-1})\, P(o_t|o_{t-1}, a_{t-1})\,. \end{aligned} \tag{4.4}$$

$\alpha(o_T)$ is the joint probability of observing $f_{1:T}$ and being in state o_T. We can similarly formulate the recursive backward part:

$$\begin{aligned} \beta(o_T) &\equiv P(f_{T+1:1}, o_{T+1:1}, a_{T+1:1})\,, \text{ and} \\ \beta(o_T) &= \sum_{t=1}^{T} \beta(o_{t+1}) P(f_{t+1}|o_{t+1})\, P(o_{t+1}|o_t, a_t)\,. \end{aligned} \tag{4.5}$$

$\beta(o_T)$ is the conditional probability of future observations $f_{T:1}$, under the assumption of being in state o_T. The posterior probability can then be obtained from

$$P(o_t|f_{1:T}, a_{1:T}) = \sum_{t=1}^{T} \alpha(o_t)\ \beta(o_t)\,. \tag{4.6}$$

Before we can make use of the forward-backward algorithm in practice, there is another important issue to be addressed. In the recursion of the forward and backward parts, we note that at each step the new value is obtained from the previous value by multiplication. These probabilities are often far less than unity and the values can go to zero exponentially fast. There are two approaches to deal with this numerical problem. One option is to use re-scaled versions of $\alpha(o_t)$ and $\beta(o_t)$ whose values remain close to unity. However, since our observation probability is in log-space, $\log(P(f_t|o_t))$, this is impractical and oftentimes not even possible. The second option is to work in log-space as well and evaluate the likelihood functions by taking the logarithms. However, because of "$\log(\sum(x)) \neq \sum(\log(x))$", we

cannot easily do that. Instead, we must use the "log-sum-exp" trick, which eventually leads to the recursive formulation of $\alpha(o_t)$ and $\beta(o_t)$ in log-space:

$$\begin{aligned}\log(\alpha(o_T)) &= \log(P(f_T|o_T)) \log \sum_{t=2}^{T} \exp(\log(\alpha(o_{t-1})) \\ &\quad + \log(P(o_t|o_{t-1}, a_{t-1}))) \,, \\ \log(\beta(o_T)) &= \log \sum_{t=1}^{T} \exp\big[\log(\beta(o_{t+1})) \\ &\quad + \log(P(f_{t+1}|o_{t+1})) + \log(P(o_{t+1}|o_t, a_t))\big] \,.\end{aligned} \tag{4.7}$$

The posterior probability is then given in log-space as

$$\log(p(o_t|f_{1:T}, a_{1:T-1}) = \sum_{t=1}^{T} \log(\alpha(o_t)) + \log(\beta(o_t)) \,. \tag{4.8}$$

4.1.3 Observation Model

For recognition, we learn a function $g : \mathbb{R}^{|\vec{f}|} \to \mathbb{N}$ that maps the extracted object features $\vec{f}$ to a unique object identifier $i \in \{1, \ldots, K\,Q\}$, each pointing to a distinct object model $o^i = \{c^k, \theta^q\}$, with $k \in \{1, \ldots, K\}$ and $q \in \{1, \ldots, Q\}$. K denotes the total number of object classes c^k and Q defines the number of discretized object poses θ^l per object class. The function is learned in a supervised fashion, with all possible features $\vec{f} = [f^1, \ldots, f^N]$ and the unique identifier i available as the value pair $(\vec{f}, i)$ at model training phase. While we assume that all features of an object are observed at training phase, we do not make this assumption during prediction, where not all components of $\vec{f}$ are observed due to feature selection.

To deal with the missing data problem at prediction time, we use a Gaussian generative model of the form

$$\log(P(f|o^i)) = \sum_{m=1}^{M} \log(\mathcal{N}(f|\mu_{o^i,m}, \sigma_{o^i,m})) \,. \tag{4.9}$$

M is the number of independent normal distributions to model a feature f. Each feature f is represented as a feature histogram with M bins, and the mean $\mu_{o^i,m}$ and variance $\sigma_{o^i,m}$ for each bin m are computed over the training data of an object with recognition state o^i. We use M independent normal distributions since a multivariate normal distribution, with full covariance matrix, would become high-dimensional for high-dimensional features. High-dimensional models are in general poorly separable as is well understood under the *curse of dimensionality*. Defining the problem as an independent Gaussian generative classifier has the advantage of handling missing features by marginalizing them out at prediction time, i.e., instead of inferring about the missing features, we simply perform

inference on a subset of the full feature vector $\vec{f}$. We can now compute the log-likelihood $\log(p(\vec{f}|o^i))$ of generating the model o^i from feature vector f as

$$\log(p(\vec{f}|o^i,\psi)) = \sum_{n=1}^{N} \psi_n \sum_{m=1}^{M} \log(\mathcal{N}(f^n|\mu_{o^i,m}, \sigma_{o^i,m})), \tag{4.10}$$

with ψ^n denoting a scaling factor to scale the different feature distributions. The feature distribution in (5.14) allows us to integrate out the marginals very easily. In practice, marginalization of a feature is simply done by setting the scaling factor to zero for all the features we do not want to include in the inference. With that, the log-likelihood $\log(P(\vec{f}|o^i,\vec{\psi}))$ is only computed over a subset of all possible features, making it independent of the dimensionality of $\vec{f}$ during training. Furthermore, the Gaussian representation of the features is compact and fast to train, since it only requires the computation of the Gaussian feature parameters $\mu_{o^i,m}$ and $\sigma_{o^i,m}$. Additionally, sequential learning as well as sequential updates of mean and variance are possible, which has the potential of adaptively adding new observations to our trained features.

Learning multiple models per object class has two distinct advantages. For one, inferring about an observation taken from a target object will not only yield the most probable object class, but also the most probable pose of the object. Second, the model trained for each object has a smaller variance if trained only for a partial view of the object. This becomes intuitively clear since an object looks in general very different for different viewing directions. And this again is reflected in the variance of the model.

4.1.4 HMM State Transitions

We define the state transition probability $p(o_t|o_{t-1}, a_{t-1})$ as the probability of a transition from recognition state o_{t-1} to state o_t given an action a_{t-1} (see Fig. 3.2). This means, how likely is the transition, if at time $t-1$ we believe to be in recognition state $o_{t-1} = \{c^k, \theta^q\}$ (i.e., a believed target object with object class c^k and orientation θ^q), then perform the motion action a_{t-1} and acquire a new observation and feature f_t, and finally believe to be in state $o_t = \{c^k, \theta^q\}$ at time t.

In order to determine the state transition probabilities, one uses the *Baum-Welch algorithm*, which uses expectation maximization to find the maximum likelihood estimate of the state transition parameters given a set of observed data. Commonly, training of such parameters is done in batch, meaning that all data needs to be available upfront. This, however, prohibits a sequential update with new data. Sequential update methods have been developed, but come with their own challenges, such as avoiding to get stuck in local minima, finding stopping criteria to reduce overfitting, or not corrupting existing knowledge. We will leave the learning of the transition probabilities for future work and use a custom-defined state transition matrix for now.

4.2 Action Selection

When studying object recognition, uncertainty in the state estimation generally leads to an increase in variance, whereas ambiguity in feature space increases the number of modes of the belief state. Consequently, to improve the recognition accuracy, we want both to reduce the variance and to reduce the number of modes toward a unimodal belief state. The root cause for multimodal belief states is the closeness of sensor data and the training models in feature space. As a direct result, the sensor data is likely to match with multiple object hypotheses, and hence a definitive object class cannot be found. In other words, certain viewing choices or combinations of features may be well explained by more than just one hypothesis.

In order to reduce uncertainty and ambiguity, we need to select actions and reason about new information in the form of newly computed features. However, not every feature or observation position, respectively, reduces the uncertainty in belief state to the same extent (and some even result in no reduction at all). Intuitively, we only want to incorporate new features that provide new information and move to new viewpoints that reduce the ambiguity by offering views of objects that are inherently different from views of objects from other object classes.

In summary, we aim at finding the sequence of actions which maximally reduces the uncertainty about our current object belief state. The action sequence should be minimal, depending on the belief state, since every action involves a cost. Finding the sequence of actions that will reduce the uncertainty is nontrivial because we need to quantify all feasible actions. Let $w \in \Omega$ be an action and Ω be the set of all feasible actions. To quantify each action w, we use the utility function $\mathcal{U} : \Omega \to \mathbb{R}$ to define the *utility score* of action w at time t as $s_t(w) = \mathcal{U}(w)$. The best next action w^*, which reduces the uncertainty of the belief state the most, is then computed as the action with the maximum utility score

$$w^* = \underset{w \in \Omega}{\operatorname{argmax}}\, s_t(w)\,. \tag{4.11}$$

Compared to global viewpoint exploration, the local exploitation of a viewpoint through local feature exploration has much lower associated cost. For that reason, we do not want to optimize over the entire action set consisting of extractable features and motion actions, given by the tuple (f, a), at once. Instead, we sequentially optimize—first over Ω_f, the discrete set of actions that are expressed as the features available in $\vec{f}$, followed by optimizing over the discrete set of all feasible motion actions Ω_a.

Regarding feature selection, the action of staying at the same viewpoint and computing a next feature f_{t+1} is obtained from (4.11) with $w \equiv (f, a)$ as $f_{t+1} = f^* = \operatorname{argmax}_{f \in \Omega_f}\, s_t(f, a)$. Here, the motion action is fixed, and in our case assumed to be the zero motion action $a = \varnothing$, i.e., $w = (f, \varnothing)$. In contrast, for viewpoint selection, the motion action to reach a new viewpoint is obtained after (4.11) by letting $w \equiv (f, a)$ and solving for $a_t = a^* = \operatorname{argmax}_{a \in \Omega_a}\, s_t(f, a)$, without yet selecting a specific f. A first new feature is then extracted at the new viewpoint. In addition, a third available action is to end the recognition process,

which applies whenever the stopping criteria (Subsection 4.2.4, and further specified throughout Sections 4.3 and Section 4.4) become true.

Our Bayesian inference framework leads naturally to an iterative information-theoretic approach for deciding on the next best action w^*. Information-theoretic formulations allow us to quantify actions in terms of information gain and uncertainty reduction, given our current belief state. Several methods from information theory have been shown to work well in action selection tasks. The more popular methods include *expected loss of entropy* (LE) (Paletta & Pinz, 2000; Borotschnig et al., 1998), *mutual information* (MI) (Atanasov et al., 2014; Denzler & Brown, 2002) and *Kullback-Leibler divergence* (KL) (Laporte & Arbel, 2006).

First we will formulate all three methods in a general way adapted to our action selection framework, followed by the specific formulations for feature selection and viewpoint selection.

4.2.1 Expected Loss of Entropy

LE, on average, measures the expected reduction in uncertainty of our belief state. More concretely, it measures the difference in uncertainty between our current posterior belief $P(o_t|f_{1:t}, a_{1:t-1})$ and the predicted posterior belief $P(o_t|f_{1:t}, a_{1:t-1}, w)$, if we were to take an action w.

The entropy H quantifies the uncertainty of a random variable. We are interested in computing the entropy of the posterior distribution $P(o_t|f_{1:t}, a_{1:t-1})$ because it expresses a measure of uncertainty (respectively certainty) about our belief. The entropy is zero if the outcome of the experiment is unambiguous and it reaches its maximum if all outcomes of the experiment are equally likely. This means, if the uncertainty is high, multiple models are likely to explain the data. Finally, the entropy over the object classes is given by

$$H(o_t|f_{1:t}, a_{1:t-1}) = -\sum_{i=1}^{KQ} P(o^i|f_{1:t}, a_{1:t-1}) \ \log P(o^i|f_{1:t}, a_{1:t-1})\,. \tag{4.12}$$

Given the definition of the entropy, we can now define the utility score $s_t(w)$ for an action w based on LE as

$$s_t(w) \ = \ \sum_{i=1}^{KQ} P(o^i|f_{1:t}, a_{1:t-1}) \ E[\Delta H(o^i, f_{1:t}, a_{1:t-1}, w)]\,, \tag{4.13}$$

with KQ object models in the model set. LE is expressed as the expected difference in entropy $E[\Delta H(\cdot)]$; it represents our expectation on the difference in uncertainty between the current and predicted posterior distribution. This expected difference is weighted by our current belief $P(o^i|f_{1:t}, a_{1:t-1})$, such that the utility score reflects the reductions in likely object beliefs. Moreover, we must compute the average LE because we cannot be certain about our belief state—hence we average over all beliefs.

The expected difference in entropy for possible actions $w \equiv (f,\, a)$ is eventually computed as

$$\begin{aligned} E[\Delta H(o^i, f_{1:t}, a_{1:t-1}, w)] \;=\; & H(o_t | f_{1:t}, a_{1:t-1}) \\ & - \int p(f|o^i, a)\, H(o_t | f_{1:t}, a_{1:t-1}, w)\, df\,. \end{aligned} \tag{4.14}$$

As we do not know which feature to expect, we need to integrate over the entire feature space. The feature distribution $p(f|o^i, a)$ is given by the observation model. Since (4.14) is not possible to compute in closed form, we approximate by sampling from the feature distribution, and obtain

$$E[\Delta H(o^i, f_{1:t}, a_{1:t-1}, w)] \;\approx\; H(o_t | f_{1:t}, a_{1:t-1}) \;-\; \sum^{\Gamma} H(o_t | f_{1:t}, a_{1:t-1}, w)\,, \tag{4.15}$$

where Γ is the number of samples drawn from the feature distribution $P(f|o^i, a)$. We can approximate (4.15) further by sampling with zero variance, which yields the mean μ_{o^i} of our trained object model o^i, and with that, a *mean feature* $f_{\mu_{o^i}}$. We eventually write

$$\begin{aligned} E[\Delta H(o^i, f_{1:t}, a_{1:t-1}, w)] \;\approx\; & H(o_t | f_{1:t}, a_{1:t-1}) \\ & - P(f_{\mu_{o^i}} | o^i, a)\, H(o_t | f_{1:t}, f_{\mu_{o^i}}, a_{1:t-1}, a)\,. \end{aligned} \tag{4.16}$$

The evaluation of $s_t(w)$ is quite costly to compute since it involves a prediction step. With each $w \in \Omega$ we want to evaluate we need to compute the sequential update of the posterior distribution given the predicted observation. Even with the mean feature approximation of (4.16), this still involves the computation of KQ predictive posteriors. Restricting attention to only the most probable o^i object hypothesis can further reduce the computational cost.

4.2.2 Mutual Information

In contrast to LE, which uses the predictive posterior distribution, MI relies on the conditional entropy to select the next action. We do not need to compute the update of the posterior distribution, which makes the evaluation cheaper and thus much faster.

The utility score is computed from the averaged MI as $s_t(w) = I(o_t; f|a)$, with $w \equiv (f,\, a)$, by

$$I(o_t; f|a) \;=\; \sum_{i=1}^{KQ} P(o^i | f_{1:t}, a_{1:t-1}) \;\int p(f|o^i, a) \;\log \frac{p(f|o^i, a)}{\sum_{j=1}^{KQ} p(f|o^j, a)} df\,. \tag{4.17}$$

Since this quantity again is not possible to compute in closed form, we approximate (4.17) with the mean feature $f_{\mu_{o^i}}$ as well, which results in

$$s_t(w) \;\approx\; \sum_{i=1}^{KQ} P(o^i | f_{1:t}, a_{1:t-1}) \; P(f_{\mu_{o^i}} | o^i, a) \; \log \frac{P(f_{\mu_{o^i}} | o^i, a)}{\sum_{j=1}^{KQ} P(f_{\mu_{o^i}} | o^j, a)}\,. \tag{4.18}$$

Generally speaking, the mutual information can be understood as the extent the uncertainty about variable o_t is reduced, when, given an action a, an observation f has been made.

4.2.3 Jeffrey's Divergence

Reducing the ambiguity between object classes is equivalent to finding viewing positions which are inherently different. To this end, we can choose new viewing positions by selecting viewpoints such that, regardless of the true object and pose under observation, the most likely distributions of the resulting measurements are predictably dissimilar. Therefore, a new viewpoint must be selected in such a way that competing hypotheses will appear as different as possible, i.e., that a maximum dissimilarity is measured by the features.

A measure of dissimilarity or "distance" between observations is the relative entropy or *Kullback-Leiber divergence*. The KL-divergence presents the amount of information lost if samples from the distribution P_1 are assumed to be samples from P_2. The K-L divergence is defined as follows:

$$KL(P_1 \parallel P_2) = \int_{-\infty}^{\infty} p_1(x) \log \frac{p_1(x)}{p_2(x)} \, dx \,. \tag{4.19}$$

However, $KL(P_1 \parallel P_2)$ is not a true distance function since it is not symmetric and does not obey the triangle inequality. We can formulate a symmetric version based on the K-L divergence, also known as *Jeffrey's Divergence* (JD), $J(P_1 \parallel P_2) = KL(P_1 \parallel P_2) + KL(P_1 \parallel P_2)$.

The evaluation of the Jeffrey's divergence is a potential computational bottleneck. In our case, the observation probability $P(f|o^i, a)$ is defined as a continuous probability density function and would require to evaluate an integral, which is not possible in closed form. Instead, we approximate the integral once more using the mean features. We evaluate the best action by computing the utility score

$$\begin{aligned} s_t(w) \approx \sum_{i=1}^{KQ} \sum_{j=1}^{KQ} & P(o^i|f_{1:t}, a_{1:t-1}) \; P(o^j|f_{1:t}, a_{1:t-1}) \\ & \cdot \; J(P(f_{\mu_{o^i}}|o^i, a) \parallel P(f_{\mu_{o^i}}|o^j, a)) \,. \end{aligned} \tag{4.20}$$

4.2.4 Convergence and Stopping Criterion

The sequential update allows to continuously acquire new information and with that reduce the uncertainty of the belief state. Unfortunately, one cannot guarantee that the sequential decision process reduces the uncertainty in every single step. But, on average, by utilizing the information-theoretic action selection, the uncertainty will be reduced.

For information-theoretic methods, it is important to find adequate *stopping criteria*. In the case of feature selection, we need a stopping criterion that defines when to stop

adding new features observed from the current viewpoint and instead rather move to a next viewpoint. Similarly, with respect to viewpoint selection, we need a stopping criterion that determines when to terminate the object recognition process and return the inference result made so far. Unfortunately, the results of an information-theoretic action selection method highly depend on the stopping criteria as well as the parameters controlling those criteria. Wrong choices can lead to sub-optimal performance, with all possible actions being processed in the worst case.

4.3 Feature Selection

Modern sensors produce vast amounts of data, which enables the computation of costly and very high-dimensional features. Each feature in our total feature set is of value, since certain objects are better explained by some features than others. This means, on average, the total feature set increases the classification accuracy. But, seldom all the features are needed to correctly classify an object because many features are correlated. Features can be highly correlated across object classes, i.e., features can be very close to each other in feature space. The result is that, if such features are combined, they do not really decrease the uncertainty of our belief state.

Feature selection is a way to reduce the amount of features we need to compute during classification. Consequently, feature selection can help to decrease the computation cost of extracting features from raw data. This means, we want to extract as few features as possible, but at the same time, we want to extract those features that reduce the uncertainty of our belief state the most. Preferably, selected features will result in a unimodal distribution.

Feature selection can be performed at training time or at prediction time. At training time, the feature space is reduced in advance by selecting relevant features, such that the loss of significant information is minimal. At prediction time, features are selected and computed online. In this work, we train our classifier with all available features, while during prediction time, we only select and infer over features that yield new information. Online feature selection increases the flexibility of the classifier. It keeps complexity low during training and facilitates updates by adding new features without the need to re-train the entire training set.

Feature selection becomes especially important on robot platforms with minimal computation power and short operation time (e.g., quadcopter robots), since it allows us to only compute relevant features. In the following subsections, we describe two methods for feature selection, one using LE and one using MI.

4.3.1 Feature Selection by Expected Loss of Entropy

In feature selection, we are interested in finding the next best feature $f_{t+1} = f^*$. For feature selection by LE, this means that LE expresses the difference in uncertainty between the current belief state and the updated belief state after adding one additional feature f.

During the selection process, we do not move the robot, so we set the motion action to the zero motion action, $a = \varnothing$. To select the next feature, we use the mean feature formulation. The action w now simply consists of selecting one mean feature $f_{\mu_{o^i}}$ we want to evaluate. $f_{\mu_{o^i}}$ is selected from the full mean feature vector $\vec{f}_{\mu_{o^i}}$ by marginalization. Since the feature selection step does not depend on the motion action a, we can neglect the term in the predictive posterior distribution. Finally, LE for feature selection is derived from (4.13) as

$$s_t(w) \approx \sum_{i=1}^{KQ} P(o^i|f_{1:t}, a_{1:t-1})\ E[\Delta H(o^i, f_{1:t}, f_{\mu_{o^i}}, a_{1:t-1})]\,, \text{ with} \tag{4.21}$$

$$\begin{aligned} E[\Delta H(o^i, f_{1:t}, f_{\mu_{o^i}}, a_{1:t-1})] = {} & H(o_t|f_{1:t}, a_{1:t-1}) \\ & - P(f_{\mu_{o^i}}|o^i)\, H(o_t|f_{1:t}, f_{\mu_{o^i}}, a_{1:t-1})\,. \end{aligned} \tag{4.22}$$

Stopping Criterion

For feature selection by LE, we stop integrating more features when the expected loss of entropy for the selected feature is small, i.e., $\max(s_t(w)) < \tau_1$, where τ_1 is a constant positive threshold parameter.

4.3.2 Feature Selection by Mutual Information

Similar to LE for feature selection, the feature selection by MI does not depend on the robot motion action either. Thus, using (4.18) and the mean feature formulation, we can compute the utility score as

$$s_t(w) \approx \sum_{i=1}^{KQ} P(o^i|f_{1:t}, a_{1:t-1})\ P(f_{\mu_{o^i}}|o^i)\ \log \frac{P(f_{\mu_{o^i}}|o^i)}{\sum_{j=1}^{KQ} P(f_{\mu_{o^i}}|o^j)}\,. \tag{4.23}$$

Stopping Criterion:

We stop to select a new feature from the current location as soon as $s_t(w)$ has decreased compared to the previous value (Verleysen & Rossi, 2009).

4.4 Viewpoint Selection

The goal in view planning is to evaluate and select new viewpoints that yield novel and useful information about the object. Initially, we have not observed anything. So every observation has equal usefulness under the assumption that we do not have any prior knowledge and a uniform belief state. However, every subsequent observation from a different viewpoint bears additional information that may help to identify the object correctly. The challenge is to evaluate potential new viewpoints which gain us new information, but also viewpoints that help us reduce the uncertainty of our belief state.

We want to find new viewpoints that reduce the ambiguity and with that reduce the modes of our belief state. In the following subsections, we formulate three methods for viewpoint selection based on three different information-theoretic measures, namely LE, MI and the KL-based Jeffrey's divergence.

4.4.1 Viewpoint Selection by Expected Loss of Entropy

Compared to the LE formulation for feature selection, we now want to move the sensor to a new observation position, thus we need to incorporate the motion action a. In addition, we must define what feature to use. We again rely on the mean features $f_{\mu_{o^i}}$. However, in contrast to the case of feature selection above, we now use the full mean feature vector $\vec{f}_{\mu_{o^i}}$ instead of only the marginals. As the full set of features performs overall on average the best, it makes sense to employ the full vector including all mean features. Since we can sample the mean features from our training models, there is no overhead in feature computation, as there would be for making an actual observation. For viewpoint selection, we thus define the action as the tuple $w \equiv (\vec{f}_{\mu_{o^i}}, a)$. Again, we do not get around to compute a complete update of the posterior distribution—this time also including the transition of the sensor from the currently believed state to a new state by performing motion action $a_t = a^*$. Based on (4.13), we can now write

$$s_t(w) \approx \sum_{i=1}^{KQ} P(o^i|f_{1:t}, a_{1:t-1})\; E[\Delta H(o^i, f_{1:t}, \vec{f}_{\mu_{o^i}}, a_{1:t-1}, a)]\,, \quad \text{with} \tag{4.24}$$

$$\begin{aligned} E[\Delta H(o^i, f_{1:t}, \vec{f}_{\mu_{o^i}}, a_{1:t-1}, a)] = {} & H(o_t|f_{1:t}, a_{1:t-1}) \\ & - P(\vec{f}_{\mu_{o^i}}|o^i, a)\, H(o_t|f_{1:t}, \vec{f}_{\mu_{o^i}}, a_{1:t-1}, a)\,. \end{aligned} \tag{4.25}$$

4.4.2 Viewpoint Selection by Mutual Information

As seen for feature selection before, the difference between MI and LE is that for MI the second term in (4.25) is the conditional entropy instead of the entropy of the posterior. From (4.18), we now derive the utility score for viewpoint selection, using MI and the full mean feature vector $\vec{f}_{\mu_{o^i}}$, as

$$s_t(w) \approx \sum_{i=1}^{KQ} P(o^i|f_{1:t}, a_{1:t})\; P(\vec{f}_{\mu_{o^i}}|o^i, a)\; \log \frac{P(\vec{f}_{\mu_{o^i}}|o^i, a)}{\sum_{j=1}^{KQ} P(\vec{f}_{\mu_{o^i}}|o^j, a)}\,. \tag{4.26}$$

The motion action a is indirectly used; the mean feature vector is generated, given a recognition state $o_t = o^i$, once the motion action a has been applied.

4.4.3 Viewpoint Selection by Jeffrey's Divergence

Similar to the MI formulation in (4.26) above, we can obtain the utility score $s_t(w)$ for the viewpoint selection by JD through the general action selection equation (4.20) from section 4.2.3, utilizing the full vector of mean features $\vec{f}_{\mu_{o^i}}$:

$$s_t(w) \approx \sum_{i=1}^{KQ} \sum_{j=1}^{KQ} P(o^i|f_{1:t}, a_{1:t-1})\ P(o^j|f_{1:t}, a_{1:t-1}) \cdot J(P(\vec{f}_{\mu_{o^i}}|o^i, a) \parallel P(\vec{f}_{\mu_{o^i}}|o^j, a))\,. \tag{4.27}$$

4.4.4 Stopping Criterion

The stopping criterion for whether to select a new viewpoint or to terminate the recognition process depends on the entropy. For all three viewpoint selection methods (Subsections 4.4.1, 4.4.2 and 4.4.3), we will stop and return the MAP estimate of the posterior distribution whenever $H(P(o_t|f_{1:t}, a_{1:t-1}) < \tau_2$. τ_2 is a second threshold parameter, which is set to a constant positive value.

4.5 Experiments

In this section, we present the results from extensive evaluations of our framework on real world data and with a quadcopter robot. All the evaluations are performed on two different sets of features. The first feature set, *simple features*, are easy and fast to compute off-the-shelf features. The second set of features, *kernel descriptors*, are more complex, generalize better, but have a higher computation cost associated. The evaluation over the two different feature sets provides an insight into recognition performance and computational cost, and demonstrates that our system is general and can handle different kinds of feature types.

We report on the performance of individual modules and then demonstrate the performance of the complete system. First, we show the performance of single-view object recognition when using the full feature set, which serves as the baseline. Our feature selection framework achieves a reduction in computation time, while the recognition accuracy remains comparable to the accuracy obtained for the baseline evaluation. Following, we evaluate the unified feature selection and viewpoint selection approach in a multi-view object recognition setting. For both feature sets, we give an intuition of the trade-off between increased recognition accuracy and additional computation cost of the multi-view approach. In addition, we test our system for its applicability to resource-constrained platforms through quadcopter experiments as well as demonstrate its usability in related tasks like object change detection.

4.5.1 RGB-D Object Dataset

We evaluated our object recognition and action selection framework on the publicly available RGB-D object dataset introduced in (Lai et al., 2011). The dataset contains 41 877 data points, of which we use roughly one-third as the test points. The dataset consists of a color image, a 3D point cloud and the orientation of the object, for a total of 300 everyday objects. For each object, the data is recorded from directions all around the object, including the three different viewing angles of 30°, 45° and 60° per direction (see Fig. 3.6 on the left and in the center). The overall dataset is challenging, since the objects in the dataset can look very similar, especially when they belong to the same object category. Furthermore, the objects in the dataset exhibit large changes in illumination due to the different viewing angles from which the objects were captured, which may make classification difficult even across different categories.

The feature vector $\vec{f}$, which is used throughout our framework, is comprised of N independent different features for each feature set. During this evaluation, we will use two different feature sets, one containing simple features, the other one containing the more complex kernel descriptors. In accordance with (5.14), each component of the feature vector is expressed as an independent normal distribution. The parameters of the independent normal distribution are learned during training. One big advantage of this representation is the possibility of sequential training. This means we can sequentially update already learned parameters or add a new model without re-training all model parameters. The scaling factor ψ^n is chosen empirical and set to {Color, Bbox, SIFT, VFH} = [1.0, 0.02, 0.2, 0.001] for the simple features. When using the Kernel descriptors we have set the scaling factor to 1.0 for all dimensions.

In all our experiments, we use the same evaluation procedure as proposed in (Lai et al., 2011) (Bo, Ren, & Fox, 2012) (Bo et al., 2011). To speedup the training process we subsample the training set to use every 5th data point. We test our framework on the task of recognizing the instance of an object. We train our object models from $\vec{f}$ using data captured from viewpoints at 30° and 60° in each direction, and test them against features computed from data that is recorded from viewpoints at 45°. Based on the computed features, we generate $Q = 8$ different object models per object instance, which clusters the viewing directions horizontally into bins spanning 45°. This subdivision trades computation cost off against performance, and defines the accuracy of the objects' pose estimates.

All the reported accuracies of our multi-view object recognition results are expressed as *recall* and are average values over 10 trials. Regarding the stopping criteria, we terminate the action selection procedure whenever either the entropy of the posterior distribution has been reduced to $\tau_2 = 0.1$ or the MAP estimate of the posterior distribution exceeds 60% in accuracy. For the LE formulation in the feature selection, we set $\tau_1 = 0.1$. After termination, we use the MAP estimate of the posterior distribution according to (4.3) to determine the most likely object.

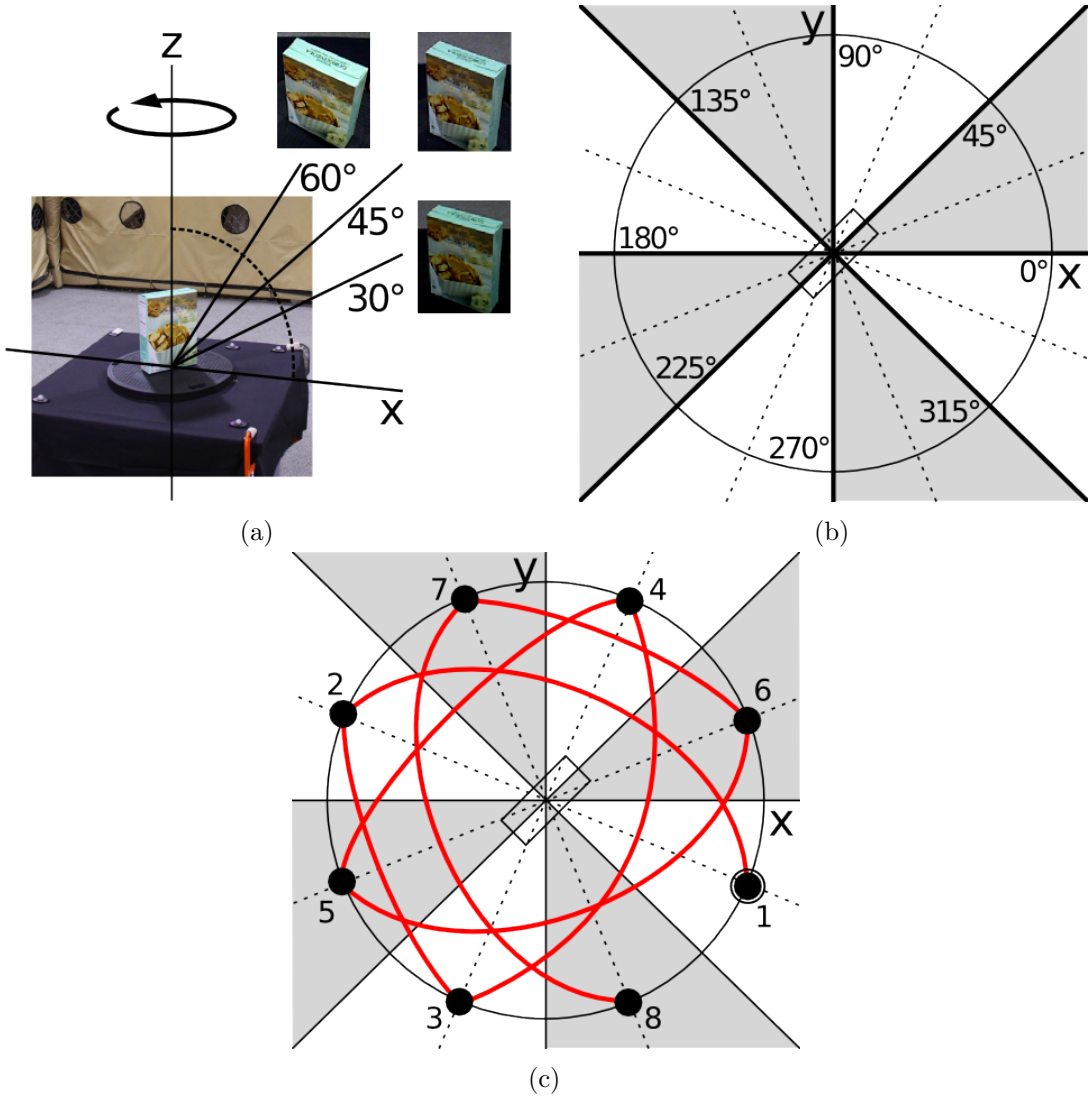


Figure 4.3: Experimental setup. (a): Each object is placed on a turntable that rotates around the z-axis (yaw rotation) of the world coordinate frame. Object data is recorded from three different viewing angles of 30°, 45° and 60° per viewing direction. (b): Viewing directions denote directions in the x-y plane of the coordinate frame from which the object can be observed. The plane is subdivided into bins of 45° around the object. Observations of the object from the same bin will lead to similar estimations of the object's pose. (c): In multi-view object recognition, a sequence of viewpoints can be planned in different ways. The shown path (red) results from the simple heuristic "motion along a star pattern", which assumes that the viewpoint opposite the current viewing direction yields the most information and thus is the one to visit next.

4.5.2 HMM State Transitions

As mentioned in (4.1.4), for this work, we have design four simple state transitions which are stored in a look-up table Tab. 4.1. The state transition probabilities are choose with the following premise. When transitioning from one belief state o_t to the next belief state o_{t+1} given a motion action a_t we want to favor a transition that stays in the same object instance or category and ends up in the correct object orientation. This allows us to incorporate the knowledge we have of how the robot has traveled when taking a motion action. Furthermore, we define transitions were the object belief would change into a different instance or category as not favorable. This achieves a constant changing of the belief state from one object to another, which occurs often in multimodal belief states. Learning the transition probabilities could certainly improve the overall performance and lead to higher recognition accuracies. However, we will leave the learning of the transition probabilities for future work, and for now use the state transitions defined as in Tab. 4.1.

Transition	Probability
Same instance, correct angle	0.4
Same instance, wrong angle	0.3
Wrong category	0.2
Wrong instance	0.1

Table 4.1: HMM state transition look-up table

4.5.3 Simple Features

We designed a set of simple feature that are fast and easy to extract from RGB-D data. We will show that simple and easy to compute feature can be very powerful in terms of small computation time and can achieve high recognition accuracy. In total we designed four different features and compute from which we create the feature vector $\vec{f}$ with $N = 4$ independent features: object bounding box, color histogram, SIFT (Lowe, 2004) and viewpoint feature histogram (VFH) (Rusu, Bradski, Thibaux, & Hsu, 2010). Each feature represents a different object property: size, color, scale-invariant image description and geometry. In accordance with (4.10), each single feature or component of a feature histogram, respectively, is expressed as independent normal distribution. As we will see, the *color feature* is the fastest and most powerful single feature among the four features, and thus the first feature we evaluate at each location.

4.5.4 RGB-D Kernel Descriptors

In contrast to our simple features, we also tested our framework on more complex and state-of-the art RGB-D kernel descriptors (Bo et al., 2011; Bo, Ren, & Fod, 2010). We use the code the authors provide to compute the kernel descriptors, which the authors

since publication of the previous two papers have improved. In total, the kernel descriptors consist of seven $N = 7$ features, each of which has the dimensionality of 1000. Fro the RGB-D data we extract three RGB features, two point cloud features and two depth features. The features are much more expressive than ours, however this comes with a price of a higher extraction cost for all features. Additionally, to compute the features, the entire training set need to be available upfront. As the first feature for our feature selection process, we extract the *RGB gradient kernel*, which has performed the best as single feature.

4.5.5 Online Feature Selection

As a start, we evaluate the usefulness of online feature selection in a single-view recognition task. We show that selecting the right subset of features can reduce the computation cost drastically, while keeping comparable recognition performance. We will show the performance and computation cost on the two feature sets. We present the evaluation for the simple features first, followed by the evaluation using the kernel descriptors.

Simple Features

Each feature has a cost associated for computation; the average computation times of our features are as follows: 0.01 s for the bounding box, 0.0005 s for the color histogram, 0.025 s for the SIFT features, and 2.0 s for the VFH. Fig. 4.4 shows the average total cost of computation per object and the recognition accuracy plotted against individual feature combinations. The color feature seems to have the best single recognition performance and is the cheapest to compute. The VFH on the other hand does not have a lot of recognition power as such and is rather expensive. From a cost vs. performance standpoint, the feature combination {Color, Bbox, SIFT} performs best with an accuracy of 68.91% and a low average feature computation cost of 0.04 s. However, note that the VFH also has its value since, if included, it increases the recognition accuracy to 74.59%, as detailed in Tab. 4.2. Additionally, we can see that using feature selection (MI,LE) reduces the cost drastically while keeping recognition accuracy similar to using all features.

Tab. 4.2 clearly shows that employing feature selection can reduce the average total cost of the feature computation considerably (see the third and fourth row of column "F Cost" in Tab. 4.2). Note that the recognition accuracy keeps similar values as if all the features were used. Here, we utilize LE and MI for feature selection according to Sec. 4.3. Among the two different feature selection methods, the LE measure seems to select more features, and more importantly, when compared to MI, is computationally much more expensive because we need to update the posterior distribution and compute a full update step.

We compared all results with the results of the original dataset in (Lai et al., 2011) and (Bo et al., 2011). The first paper has very comparable results to our HMM framework and uses similar features to ours. No timing has been provided but the computation cost is expected to be equal to (or slightly higher than) the cost of our features. The latter

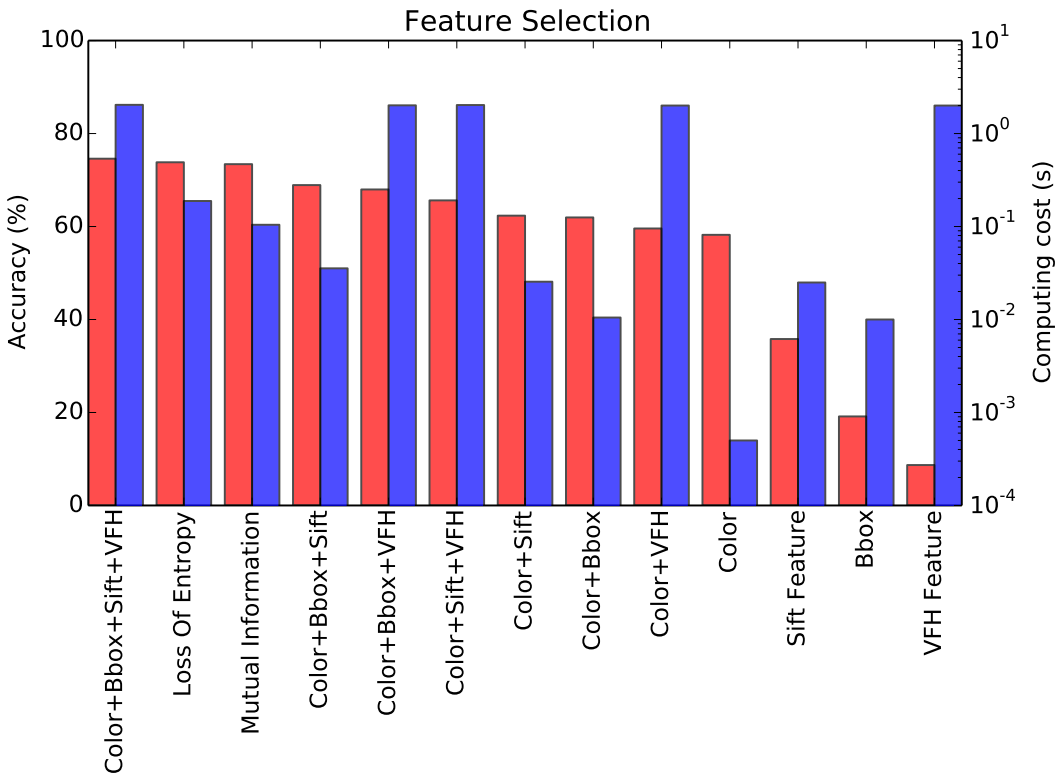


Figure 4.4: Simple features—Performance and cost of different feature combinations in single-view object recognition. The recognition accuracies are displayed as red bars (left side scale, in %). The average cost of computing the feature combinations per object is shown in log scale as blue bars (right side scale, in seconds).

paper reports much higher recognition accuracies; this is mainly due to a more descriptive feature which seems to generalize very well (also refer to the following subsection). The cost for such a feature, however, is a very high computation time. Moreover, the features used in the two papers are learned in batch, which makes them less flexible compared to our independent feature learning approach.

RGB-D Kernel Descriptors

Similar to the simple features, we evaluate the performance of our feature selection framework when used together with the kernel descriptors. In Tab. 4.3, we show the accuracy and cost for single-view object recognition in combination with feature selection. The first row of the table shows the recognition performance of the kernel descriptor with our Bayesian framework. We can see that compared to the linear SVM classifier used

Method	Accuracy	F Cost (s)	FS Cost (s)	Total (s)
C+B+S	68.91%	0.04	0.00	0.04
C+B+S+V	74.59%	2.04	0.00	2.04
Loss Entropy	73.82%	0.19	1.05	1.24
Mutual Information	73.21%	0.1	0.2	0.3
(Lai et al., 2011) (kSVM)	74.80%	-	-	-
(Bo et al., 2011) (Kernel Descriptor)	91.2%	3.2	0.0	3.2

Table 4.2: Simple Feature — Individual performance for single-view object recognition. "Loss Entropy" and "Mutual Information" denote the feature selection method, with "F Cost" being the average cost per object for computing the features and "FS Cost" being the average cost per object for feature selection.

Method	Accuracy	F Cost (s)	FS Cost (s)	Total (s)
Kernel Descriptor	90.3%	3.2	0.00	3.2
Loss Entropy	89.94%	0.88	1.71	2.59
Mutual Information	90.61%	0.83	0.46	1.29
(Bo et al., 2011) (Kernel Descriptor)	91.2%	3.2	0.00	3.2

Table 4.3: Kernel Descriptors — Individual performance for single-view object recognition. "Loss Entropy" and "Mutual Information" denote the feature selection method, with "F Cost" being the average cost per object for computing the features and "FS Cost" being the average cost per object for feature selection.

in (Bo et al., 2011), our accuracies are slightly lower with 90.3% vs. 91.2%, respectively. Furthermore, we can see that feature selection using the LE formulation, compared to the MI formulation, is still significantly slower due to the increased computational overhead. Both approaches, however, show a significant reduction in the average computation time of the features, meaning significantly less time is spent extracting features, when the feature selection is used. For the MI formulation, for example, we achieve a reduction from 3.2 s to 0.83 s. The MI formulation arrives at the lowest total cost of 1.29 s for feature computation and feature selection. This is a reduction of a factor of 2.5 in total computation time. Even though we substantially reduce the cost, and with that the amount of features to extract, the recognition accuracy stays comparable to the case where the full feature set is used. The MI approach achieves 90.61% in accuracy.

Discussion

Overall, we can see that feature selection reduces the set of features to extract—and thus the computation cost—drastically. The result that the accuracy is comparable, or in

some cases even slightly better than when using the full feature set, further validates the usefulness of our approach.

The feature selection in the case of simple features reduces the cost of feature computation the most. From Fig. 4.4 we see that the VFH feature does not hold a lot of discriminative information for most of the objects. This is reflected in the large reduction in computation time, since most of the times, the VFH feature is not computed, hence the low average feature computation time of 0.1 seconds.

We have done some further investigation about the behavior and performance of the VFH features. One major reason for the poor performance regarding this dataset is the object distribution. Compared to previously tested objects, this dataset contains a lot of similar objects of the same category. For instance, the dataset contains an ?apple? category with different kinds of apples. Geometrically, the apples in the dataset are all very similar only distinguishing themselves slightly in size but more in color. During instance classification, many objects are misclassified which results in a very poor classification performance. We tested the performance of the VFH features on this dataset with category classification and the performance is much higher. No doubt the performance of the VFH features, compared to previous experiments is influenced by not using a SVM for classification.

Similar to the simple features, we can also see a reduction of the average feature computation time for the more complex kernel descriptors. However, here the reduction is not as pronounced. More features are useful, but more importantly, there is not one single feature that has, compared to all the others, a large computation cost, which can be saved.

The average computation cost for computing the next informative feature is higher for the kernel descriptors. The kernel descriptors are of higher dimensionality in total, which slightly increases the computation time. In contrast to the simple features, the total average cost of the kernel descriptors is about a factor of four higher. However, this increase in computation cost is well justified, since it is accompanied by a large increase of 15% in recognition accuracy.

4.5.6 Multi-View Object Recognition

We now present the performance of our unified approach of feature and viewpoint selection. We evaluate the performance for different view planning methods; in total, we compare five different ways to estimate the next viewpoint: *Random*, *Loss of Entropy*, *Mutual Information*, *Jeffrey's Divergence* as well as with a simple *Heuristic*. The simple heuristic serves for comparison with our information-theoretic approaches and is based on the assumption that viewpoints opposite the current viewing directions yield the most information. Intuitively, the sensor then moves in a "star pattern" around the target object to acquire new information from the opposite side, as shown by the path in Fig. 3.6 on the right.

Method	Accuracy	Observations	Avg. Bin Error
C+B+S+V (no FS)	84.48% ± 1.4	3.4 ± 0.2	2.41 ± 0.07 (bins) (±67.5°)
Random	86.76% ± 0.4	3.2 ± 0.2	2.19 ± 0.02 (bins) (±67.5°)
Heuristic (Star)	87.16% ± 0.6	3.1 ± 0.3	2.28 ± 0.02 (bins) (±67.5°)
Jeffrey's Divergence	88.34% ± 0.6	2.7 ± 0.1	1.62 ± 0.01 (bins) (±45.0°)
Loss Entropy	89.82% ± 1.3	2.4 ± 0.2	1.54 ± 0.01 (bins) (±45.0°)
Mutual Information	91.87% ± 0.7	2.5 ± 0.1	1.48 ± 0.01 (bins) (±45.0°)

Table 4.4: Simple Features — Performance of our multi-view object recognition framework. All but the first row uses the MI measure in the feature selection method. We show the average recognition accuracy, the average number of observations and the average bin error of estimated viewing direction. Each bin represents a range of observation angles of 45°.

simple Features

Although the computation time can be substantially reduced through feature selection, the achievable recognition accuracies are limited around 75% for single-view object recognition. The reasons are the relative simplicity of our four feature types and their limitation in generalizing well for the given RGB-D dataset. However, even with fairly simple features, we can achieve improved recognition accuracies for both object class and pose by utilizing an information-theoretic multi-view object recognition approach. In the following tests, we use the MI measure for feature selection, since it is on par with the LE formulation regarding recognition performance, but the computational cost (see Tab. 4.2) is significantly lower.

Tab. 4.4 shows the performance and Tab. 4.5 summarizes the cost of multi-view object recognition. The first row shows that using random viewpoint selection with all features (i.e., no feature selection) has a very high cost but already increases the recognition accuracy by about 10%. In the second row, we can see a reduction in computation time and an increase in accuracy due to MI-based feature selection and random viewpoint selection. On average, random viewpoint selection terminates after a little more than 3 observations. The simple heuristic performs a little better than random selection, but it similarly needs on average about 3 observations to achieve a comparable accuracy. The computation time is also very similar. However, the simple heuristic does not perform as well as the information-theoretic methods. By using the information-theoretic measures for viewpoint selection, we find that LE, MI and JD all perform similarly well—with MI being the best in terms of recognition accuracy. The lowest computation time is achieved by using JD because it can compute new viewing positions in almost no time. The MI formulation is superior to the LE formulation in speed, since it does not need to compute the predictive posterior.

Method	F Cost (s)	FS Cost (s)	VS Cost (s)	Total Cost (s)
C+B+S+V (no FS)	7.02 ± 0.42	0.00	0.0	7.02 ± 0.42
Random	0.40 ± 0.04	0.83 ± 0.05	0.0	1.23 ± 0.09
Heuristic (Star)	0.45 ± 0.08	0.79 ± 0.04	0.0	1.24 ± 0.12
Jeffrey's Divergence	0.30 ± 0.03	0.58 ± 0.02	3.E-4	0.88 ± 0.05
Loss Entropy	0.23 ± 0.04	0.50 ± 0.04	5.4 ± 0.6	6.13 ± 0.68
Mutual Information	0.24 ± 0.03	0.51 ± 0.03	1.1 ± 0.2	1.85 ± 0.26

Table 4.5: Simple Features — Average computation cost per object for our multi-view object recognition framework. The columns show the average cost for computing the feature ("F Cost"), average cost for feature selection ("FS Cost") and average cost for viewpoint selection ("VS Cost").

We further evaluate our multi-view approach for estimating an object's pose. As previously mentioned, we train each object with a number of different object models, in our case $Q = 8$ models per object. When observing a feature, we can infer from which model it has most likely been generated, and by this, further infer about the orientation of the object. The performance of the object's pose estimation is presented in Tab. 4.4. The table shows the average bin error of the estimated viewing direction, which is computed from the difference between the inferred and the true observation angle. For the first row, all the features were computed (i.e., no feature selection) and the viewpoints were selected at random. All the other experiments use the MI formulation to select the informative features at each given observation position. On average, using random and heuristic viewpoint selection approaches, we are > 2 bins off in our pose estimation, which corresponds to an average pose error of about 67.5°. If we apply one of the three more sophisticated viewpoint selection methods, we yield an average bin error of < 2 bins. This results in an average pose error of 45°. For comparison, the experiments in (Bo et al., 2012) report an average pose error of 44.8°, which is on par with our results.

RGB-D Kernel Descriptors

Similar to the simple features, we evaluate the kernel descriptors with our unified feature and viewpoint selection framework. Tab. 4.6 and Tab. 4.7 summarize the performance of accuracy and cost, respectively. From Tab. 4.6, we see that, by using a multi-view approach with random viewpoint selection but no feature selection, the recognition accuracy, has increased from 90.61% to 95.01%, with 1.8 observations on average (first row). Furthermore, we can see that random and heuristic viewpoint selection perform almost identical, still the heuristic achieves a slightly higher accuracy but also needs on average slightly more observations. In contrast, the information-theoretic viewpoint selection methods perform again better as the simple ones, with the best result achieved by the MI formulation. The MI formulation achieves a respectable recognition accuracy of 98.14% and on average

Method	Accuracy	Observations	Avg. Bin Error
Kernel Descriptors*	95.01% ± 0.8	1.9 ± 0.1	1.95 ± 0.06 (bins) (±45.0°)
Random	95.16% ± 0.6	1.9 ± 0.3	1.85 ± 0.03 (bins) (±45.0°)
Heuristic (Star)	95.83% ± 0.4	2.0 ± 0.1	1.88 ± 0.03 (bins) (±45.0°)
Jeffrey's Divergence	96.32% ± 0.5	1.5 ± 0.2	1.53 ± 0.01 (bins) (±45.0°)
Loss Entropy	97.98% ± 1.0	1.4 ± 0.3	1.41 ± 0.01 (bins) (±45.0°)
Mutual Information	98.14% ± 0.4	1.4 ± 0.2	1.43 ± 0.02 (bins) (±45.0°)

Table 4.6: Kernel Descriptors — Performance of our multi-view object recognition framework. All but the first row uses the MI measure in the feature selection method. We show the average recognition accuracy, the average number of observations and the average bin error of estimated viewing direction. Each bin represents a range of observation angles of 45°.

* (No feature selection)

needs about 1.4 observations. Interestingly, the kernel descriptors reach similar results in terms of pose estimation as the simple features. We think the reason is that objects still look very similar when moving the observation position within 45°, making it hard to classify a target object as the exact object (i.e., correct object class and orientation angle) the model was created from.

Additionally, we also evaluated the computation cost of individual methods and modules, as can be seen in Tab.4.7. Due to its high computation cost, we can see, that the LE formulation is by far the most costly, making it again a poor choice—especially, when compared to the other information-theoretic approaches. The MI formulation achieves the highest accuracy and requires the lowest number of observations, however, the cost is higher than for the random, heuristic or JD formulation. Overall, the fastest method is the method that is based on JD; it only requires 2.36 s of computation time.

Discussion

With both feature sets, we can clearly see that having the option of taking additional observations increases the recognition accuracy. In the case of the simple features, the increase is over 15%. With about 8%, the kernel features do not experience such a large gain, however, the single-view recognition rate has here already been relatively high.

Information does not automatically mean useful information—this can especially be seen when we compare the heuristic with the MI-based viewpoint selection approach. Adding new information does not necessarily add valuable knowledge to find the correct object class. In terms of information content, the heuristic should add the most information, since we move the sensor to the complete opposite sides. Instead, when positions are chosen in terms of uncertainty reduction, which also reduces object ambiguity, better performance can be observed.

Method	F Cost (s)	FS Cost (s)	VS Cost (s)	Total Cost (s)
Kernel Descriptors*	6.11 ± 0.31	0.0	0.0	6.11 ± 0.31
Random	2.51 ± 0.16	0.94 ± 0.10	0.0	3.45 ± 0.26
Heuristic (Star)	2.56 ± 0.11	0.98 ± 0.08	0.0	3.54 ± 0.19
Jeffrey's Divergence	1.64 ± 0.09	0.69 ± 0.02	0.03 ± 0.05	2.36 ± 0.16
Loss Entropy	1.43 ± 0.1	0.67 ± 0.04	8.1 ± 0.5	10.02±0.54
Mutual Information	1.45 ± 0.13	0.68 ± 0.03	1.9 ± 0.4	4.03 ± 0.56

Table 4.7: Kernel Descriptors — Average computation cost per object for our multi-view object recognition framework. The columns show the average cost for computing the feature ("F Cost"), average cost for feature selection ("FS Cost") and average cost for viewpoint selection ("VS Cost").

* (No feature selection)

In comparison to the single-view approach, the computation time is increased due to computing the next viewpoint—but also due to travel time of the robot. Depending on the robot platform additional costs may be associated with an action like motion planning or motion risk. Currently we do not incorporate any extra costs associated with motion, however we think that could be an interesting future research direction.

For the simple feature, on average, we need to take 1–2 additional views for an improved accuracy. When using the kernel descriptors instead, it suffices to take none or only one additional observation on average. However, the computation cost for the kernel descriptors is about 2.7 times higher (when using the JD measure) compared to the simple features.

The JD formulation is by far the fastest, i.e., a factor of about 1.7 times faster compared to MI in finding a new viewpoint. In contrast, the MI method achieves about 2% better accuracy and needs on average less observations. Finally, view planning with the MI formulation seems to pick better observation positions, compared to the JD formulation, trading off accuracy for speed.

For many robotic applications, for instance manipulation, the estimated object pose estimates are not precise enough. However, the estimation of the pose within a recognition framework is of great value, because in a sense we get this information for free. For State-of-the-art object pose estimation methods to work, a rough initialization of the object pose is required. The values estimated in this work are well suited as an initial pose for refinement methods like ICP.

4.5.7 Quadcopter Experiments

In order to show the overall performance of our active multi-view object recognition framework, we have conducted experiments on our custom-built quadcopter platform, shown in Fig. 4.5.

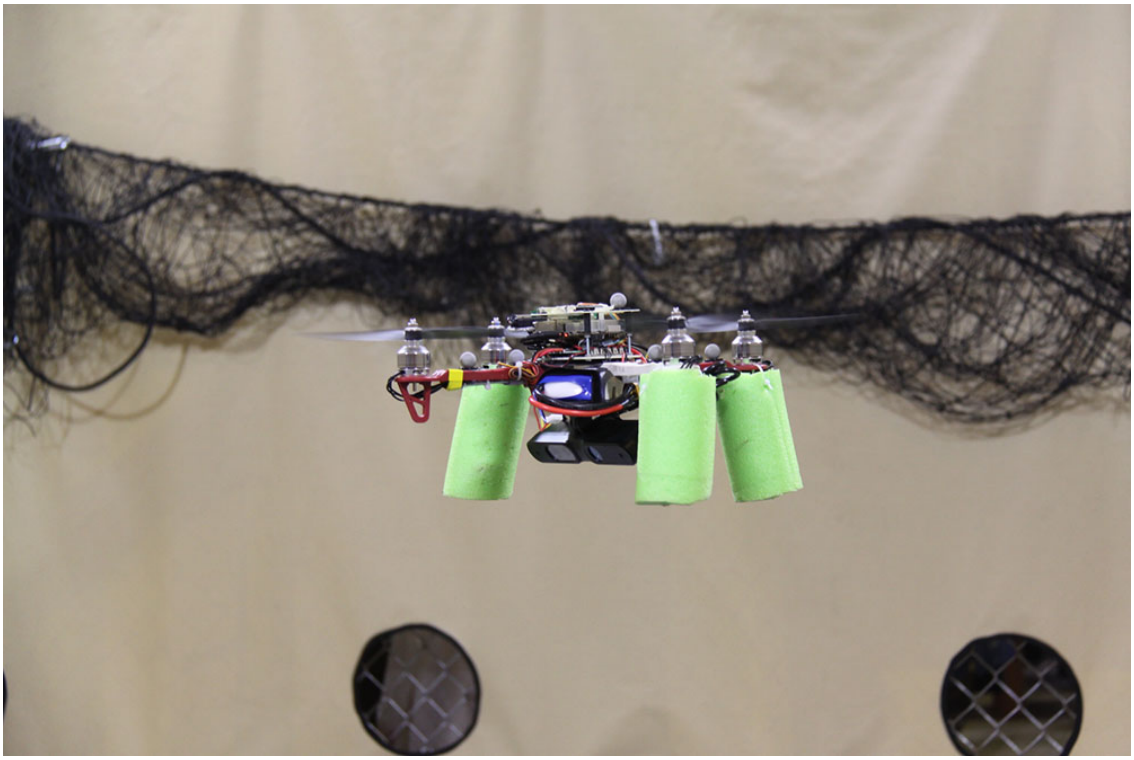


Figure 4.5: Shown is our custom quadcopter platform. The platform is build upon a glass fiber 330mm mini quad frame. In the middle of the frame we have built a superstructure that houses the sensors (e.g. IMU) and processing unit. As the onboard computer we are using a Hardkernel Odroid U3, which is a 1.7 GHz Cortex-A9 Quad-core processor with 2 GByte RAM. The onboard computer is used as the flight control hardware, running the flight control stack for tasks like attitude estimation, stabilization and waypoint execution. The custom platform allows for a variable motor and battery size, enabling to trade-off between flight-time and payload. In the current configuration, as shown in the picture, the quadcopter is equipped with an ASUS Xtion RGB-D camera, used for data collection. In the above described configuration the platform has a flight-time of about 13 minutes. To accurately estimate the position of the quadcopter indoors we use a VICON motion capture system.

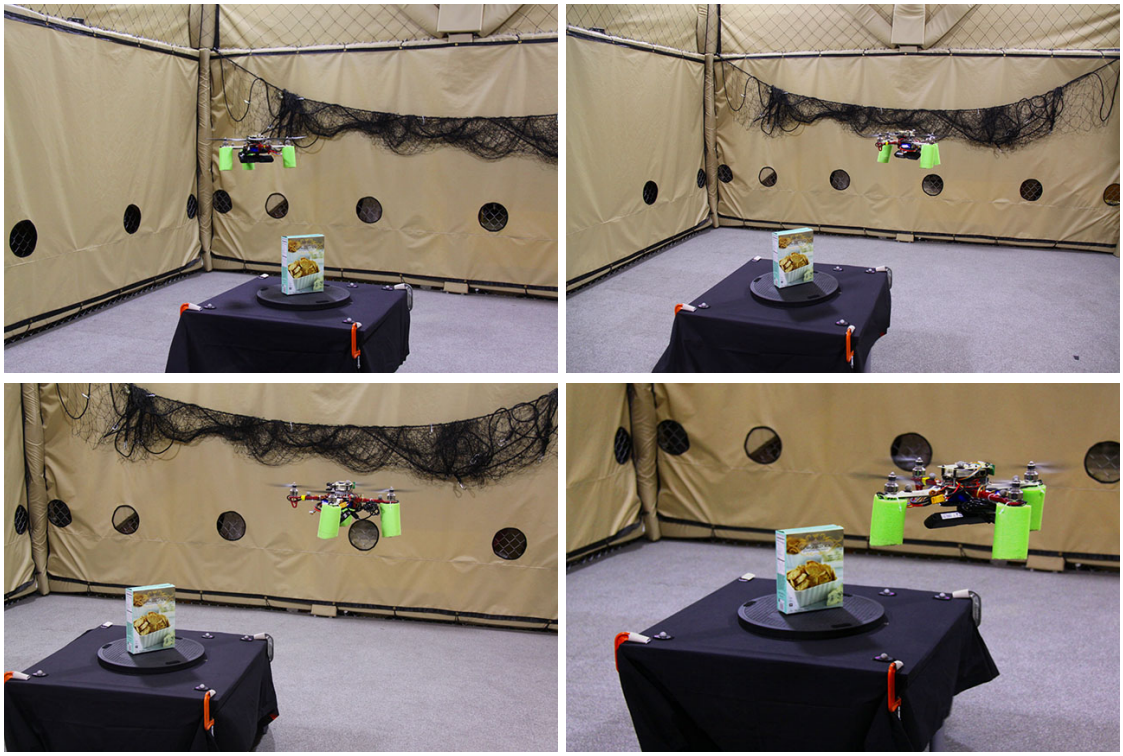


Figure 4.6: Quadcopter experiment and test setup. The quadcopter robot, equipped with a RGB-D camera, collects multiple views of a target object (e.g., a cereal box) in an active object recognition task.

As shown in Fig. 4.6, the quadcopter is equipped with an ASUS Xtion RGB-D camera for data acquisition and can fly autonomously to a new observation position. Given a target object, the task is to fly the quadcopter around the object and take as many observations as necessary to correctly recognize the object. Compared to the simulated experiments above, where data is captured with an RGB-D camera mounted on a stable tripod, experiments on a robotic platform are generally more challenging. Specific challenges regarding object recognition on a fast moving platform include motion blur and constant changes in illumination.

As we do not have any of the objects used in the RGB-D dataset at hand, we captured 10 new objects and added them to the dataset, which now consists of a total of 310 objects. We captured the data in the same way as described in (Lai et al., 2011) and chose objects similar to the ones already contained in the dataset. We trained our models with all 310 objects and the two viewing directions of 30° and 60°. In the actual experiment, the quadcopter then flies around the object at a distance and height that allows for a viewing angle of 45° roughly.

Method	Accuracy	Obsv.	Avg. Bin Error	Avg. Cost (s)
Single-View				
C+B+S+V*	82.16%	1	2.1(bins)(±67.5°)	2.14
Feature Selection (MI)	87.32%	1	1.9(bins)(±45°)	0.19
Multi-View				
Mutual Information*	96.5% ± 0.5	2.0 ± 0.1	1.6 ± 0.1(bins)(±45°)	5.63 ± 0.3
Mutual Information	99.8% ± 0.1	1.6 ± 0.01	1.1 ± 0.1(bins)(±45°)	1.48 ± 0.2

Table 4.8: Simple Features — Single-view and multi-view object recognition based on data from the quadcopter robot. In both cases, results with and without feature selection are presented. Advantages of online feature selection as well as multi-view object recognition are revealed by the experiments.
* (No feature selection)

In each run of the experiments, the robot starts at a random position with respect to the target object, from where it acquires the first observation of the object. From there on, it selects the following actions as described in Sections 4.2, 4.3 and 4.4—either inferring about another feature from $\vec{f}$ or flying to a new viewpoint. We terminate the process once our stopping criteria are reached. Fig. 4.6 shows an example of the quadcopter robot taking observations of a cereal box object during one of the experiments.

Simple Features

In Tab. 4.8 we summarizes the experimental results from the quadcopter experiments, averaged over eight runs with different starting locations. We can see that for all experiments, whether conducted with single-view or multi-view object recognition, online feature selection not only helps to decrease the computation time but also increase the recognition accuracy. The average pose error is similar to the error in the simulation experiments, however, the average bin error is smaller, which results in more correctly classified bins in total.

Multi-view recognition on average only required 1.6 observations, which means that often we are able to make a confident decision after only one observation. In some instances, however, by acquiring additional observations, the recognition accuracy could be increased by more than 10% to totally 99.8%. This, of course, comes at the price of actually moving the robot to the new viewpoint, which in turn increases the cost of recognizing the object. Whether the extra cost of moving is justified in practice, finally depends on the robot platform used, the surrounding environment, as well as the trade-off between desired recognition accuracy and energy savings. With respect to online feature selection, we can clearly see that the adaptive selection of features offers huge benefits by decreasing the computation time, which is especially beneficial for vehicles with low energy budgets like quadcopters.

Method	Accuracy	Obsv.	Avg. Bin Error	Avg. Cost (s)
Single-View				
(Bo et al., 2011) (Kernel Descriptor)	94.27%	1	1.5(bins)(±45.0°)	3.2
Kernel Descriptor*	93.67%	1	1.6(bins)(±45.0°)	3.2
Feature Selection (MI)	94.05%	1	1.5(bins)(±45°)	1.7
Multi-View				
Mutual Information *	100.0%	1.3 ± 0.05	1.3±0.03(bins)(±45.0°)	9.42 ± 0.5
Mutual Information	100.0%	1.2 ±0.02	0.7±0.2(bins)(±22.5°)	3.63 ± 0.3

Table 4.9: Kernel Descriptors — Single-view and multi-view object recognition based on data from the quadcopter robot. In both cases, results with and without feature selection are presented. Advantages of online feature selection as well as multi-view object recognition are revealed by the experiments.
* (No feature selection)

Kernel Descriptors

We present results of the quadcopter experiments using the kernel descriptors in Tab. 4.9. Again, the kernel descriptors generalize much better than the simple features, achieving a much higher recognition accuracy in single-view object recognition. However, the cost of recognizing the target object is much higher compared to the simple features. The MI-based approach for multi-view object recognition achieves 100% recognition accuracy with on average 1.2 observations needed. It seems that the pose estimation has also improved by the more complex features, with the best result achieving an average angle error of 22.5° only. Basically, we can see that the recognition result of the single-view approach is already very good. The multi-view approach can be seen as a complement for these experiments—often, it only takes one view to recognize the object, at other times, there is a benefit from taking more than one observation.

Discussion

The experiments in this subsection show our multi-view system applied to a quadcopter to perform information gathering. First of all, the experiments confirm the practicability of our approach regarding the real application on a robot platform. The accuracy presented for these experiments will probably not hold in general. Although we added 10 objects to the database, which we tried to pick, such that they were similar and of the same categories as the objects already contained in the database, they seem to be different enough to show a rather drastic increase in recognition accuracy. Therefore, the accuracy results need to be taken with some caution. Nevertheless, what the system does show is the advantages of a multi-view system in combination with online feature selection. We again achieve an increase in accuracy when given the chance of taking additional observations. Furthermore, we see a reduction in computation time thanks to the online feature selection framework. Given the limited processing capabilities of a quadcopter, these are very promising results.

With regard to energy consumption and cost to move, quadcopters are somewhat special platforms. To be able to hover or fly, the energy a quadcopter consumes is in general quite high. Compared to other robots which do not consume a lot of battery power when stationary (e.g., ground or aquatic robots), for quadcopters, moving to a new position by flying does not cost that much more energy than staying in place by hovering. This makes an active multi-view object recognition system, as the one we have presented in this paper, particularly suitable for quadcopter robots.

4.5.8 Object Change Detection

As a final application, we now show how to utilize our framework for detecting changes in the environment. Given a prior map of the environment and an object database, we want to detect if objects in the environment have been replaced, taken away or changed their pose. We use our generative object model to express the probability $P(o^i|f)$ that the object has been generated by the feature f. Given an expected object model $\hat{o}^i$ (prior information), we can compute the probability $P(u)$ that the object has changed with $P(u) = 1 - P(\hat{o}^i|f)$. If the probability $P(u) > \tau$, with $\tau \in [0, 1]$, we know that the feature observed cannot be generated by the model $\hat{o}^i$, hence either object class or object pose has changed. This can also occur when the feature does not match the model anymore due to changes in lighting or noise; in either case we need to perform further observations to arrive at a definitive conclusion about the change. If we are uncertain about the object after our first observation, we compute new viewpoints that allow for observations that are unique in terms of feature space. We acquire new observations, evaluate additional features and compute new likelihoods of the features given the expected object model $\hat{o}^i$.

In the following, we explore three different scenarios, when 1) the object has not changed, 2) the object has changed completely (i.e., it belongs to a different object class), or 3) the object has changed its orientation only (i.e., it belongs to the same class but its rotation differs in yaw). All experiments are conducted with threshold $\tau = 0.6$, meaning that we need to have a high enough certainty of the object being changed. All experiments are conducted with feature selection by MI, and then evaluated for the different view planning methods.

No Object Change

No change in object and pose refers to the scenario where the object and pose remain the same as the reference object given our prior map knowledge. This means, we want to validate whether the measurement(s) we take of the currently observed object conforms with the object we expect to see. Knowing which object we expect to see makes the recognition or validation task somewhat easier since observations need to conform very strongly with the expected object model. Similarly, when selecting new viewpoints, the expected feature at a new observation position will be very similar to the actual measurement. This makes it easier to select new viewpoints, which leads to observations that are very dissimilar to observations of other likely object candidates.

Method	Accuracy	Observations	Avg. Cost (s)
Random	91.19 ± 0.19	2.2 ± 0.18	1.02 ± 0.36
Heuristic (Star)	92.82 ± 0.26	2.1 ± 0.21	0.91 ± 0.49
Jeffrey's Divergence	96.13 ± 0.62	1.4 ± 0.12	0.54 ± 0.27
Loss Entropy	96.71 ± 0.48	1.3 ± 0.2	3.43 ± 0.49
Mutual Information	98.33 ± 0.53	1.3 ± 0.13	0.83 ± 0.38

Table 4.10: Simple features—Scenario where the object has not changed compared to the prior knowledge. The table shows the detection accuracy, number of observations needed as well as the total average computation cost in seconds.

Method	Accuracy	Observations	Avg. Cost (s)
Random	95.92 ± 0.11	1.9 ± 0.12	3.3 ± 0.47
Heuristic (Star)	96.83 ± 0.15	1.8 ± 0.14	3.2 ± 0.51
Jeffrey's Divergence	99.15 ± 0.17	1.3 ± 0.23	2.15 ± 0.27
Loss Entropy	99.81 ± 0.21	1.2 ± 0.19	9.01 ± 0.26
Mutual Information	99.74 ± 0.24	1.2 ± 0.17	3.8 ± 0.23

Table 4.11: Kernel descriptors—Scenario where the object has not changed compared to the prior knowledge. The table shows the detection accuracy, number of observations needed as well as the total average computation cost in seconds.

In Tab. 4.10 and Tab. 4.11, we show the results for the case of no change in the environment, both for the simple features and the kernel descriptors, respectively. In the two tables, we can see that both feature sets perform very similar with respect to accuracy of detecting no change and number of required observations. Both feature representations achieve on average accuracies of more than 98% and need 1.3 or fewer observations. However, we can see that the simple features are by far less expensive to compute, with 0.83 seconds for the simple features compared to 3.8 seconds for the kernel descriptors. The high accuracies show that the task of validating the measurements with respect to the expected object model is easier compared to the task of pure object recognition without any prior knowledge.

Object Change

In this scenario, the object and pose have changed compared to the map previously acquired. The task here is to detect that the target object has indeed changed compared to the expected object model. Compared to the object recognition results, this task is more difficult due to the stricter criteria for correct estimation of change. In the object recognition setting, for correct recognition, we require the MAP estimate to be the correct

Method	Accuracy	Observations	Avg. Cost (s)
Random	80.24 ± 0.25	2.4 ± 0.18	0.98 ± 0.37
Heuristic (Star)	81.71 ± 0.28	2.5 ± 0.21	1.12 ± 0.32
Jeffrey's Divergence	86.25 ± 0.36	1.9 ± 0.12	0.78 ± 0.23
Loss Entropy	89.25 ± 0.27	1.7 ± 0.2	3.03 ± 0.43
Mutual Information	88.95 ± 0.31	1.8 ± 0.13	1.16 ± 0.29

Table 4.12: Simple features—Scenario where both the object and pose have changed compared to the prior knowledge. The table shows the detection accuracy, number of observations needed as well as the total average computation cost in seconds.

Method	Accuracy	Observations	Avg. Cost (s)
Random	90.84 ± 0.25	2.2 ± 0.21	3.72 ± 0.53
Heuristic (Star)	89.75 ± 0.28	2.1 ± 0.19	3.61 ± 0.46
Jeffrey's Divergence	92.68 ± 0.36	1.8 ± 0.26	3.01 ± 0.36
Loss Entropy	94.91 ± 0.27	1.5 ± 0.16	10.97 ± 0.41
Mutual Information	93.17 ± 0.31	1.4 ± 0.21	4.12 ± 0.29

Table 4.13: Kernel descriptors—Scenario where both the object and pose have changed compared to the prior knowledge. The table shows the detection accuracy, number of observations needed as well as the total average computation cost in seconds.

object. In contrast, here, to be correctly classified as change, we require the probability $P(u) > \tau$. This means, objects that look very similar and are only observed from similar positions will also have similar likelihoods. The result is that $P(u)$ will in general not be greater than τ. Finding new observation positions with object beliefs that are inherently different from other object beliefs is equally difficult as for the object recognition task, since the prior knowledge does not provide us with any further knowledge about the current object we observe. And with that, the expected features when moving the sensor might be different from the features actually measured after moving the sensor.

In Tab. 4.12 and Tab. 4.13, we show again results for both feature representations. We can see that both accuracies are lower compared to the object recognition task in Section 4.5.6 due to the stricter recognition criteria. On average, we achieve correct change classification for 88.95% of the objects using the simple features and for 93.17% of the objects using the kernel descriptors. Again, similar to the object recognition task, the kernel descriptors seem to be more descriptive, achieving better accuracy, with the trade-off of a higher cost. The computation cost of the kernel descriptors is by about a factor of four higher compared to the simple features.

Method	Accuracy	Observations	Avg. Cost (s)
Random	87.15 ± 0.31	2.7 ± 0.23	1.31 ± 0.29
Heuristic (Star)	88.68 ± 0.36	2.6 ± 0.16	1.21 ± 0.36
Jeffrey's Divergence	95.97 ± 0.25	1.7 ± 0.15	0.61 ± 0.19
Loss Entropy	97.75 ± 0.14	1.5 ± 0.11	2.72 ± 0.31
Mutual Information	97.12 ± 0.18	1.6 ± 0.16	0.91 ± 0.21

Table 4.14: Simple features—Scenario where only the object's pose (yaw angle) has changed. The table shows the detection accuracy, number of observations needed as well as the total average computation cost in seconds.

Pose Change

In the third scenario only the object pose changes, while keeping the object the same. In our case, this means the object has rotated around its z-axis (yaw). Given our rough object pose estimation based on model training for different viewing directions, we are only able to estimate pose changes that exceed the angle discretization. In our case, we can possibly detect pose changes that are greater than 22.5° in yaw angle. A correct classification is made when the probability $P(u) > \tau$, indicating the object has changed its orientation.

Compared to the previous case, where the entire object has changed, a pose change is again easier to detect, since the expected object corresponds to the object we are observing. This in turn simplifies the task of finding new viewpoints that are different in feature space from other likely candidates. This, of course, is only a problem for objects that look very similar to other objects (in feature space) contained in the database.

In Tab. 4.14 and Tab. 4.15, we show results for the simple features and kernel descriptors. We can see that the accuracy results for the two feature representations are fairly similar with 97.12% for simple features and 98.11% for kernel descriptors, respectively. The kernel descriptors need on average 0.2 observations less but again have a higher computation time of about a factor of four.

Quadcopter Experiments

Similar to the quadcopter experiments in Section 4.5.7, we also evaluated the change detection framework on our quadcopter platform. The experimental setup remains the same as with the previous quadcopter experiments. However, instead of recognizing a target object, we are interested in detecting any change of the object given prior information. This again means, we want to detect whether the object remains unchanged, the complete object has changed, or only the pose of the object has changed.

The experiments conducted make use of the MI formulation for feature selection as well as viewpoint selection, with all parameters remaining set as before. In Tab. 4.16, we show the averaged results for the detection accuracy, number of observations needed

Method	Accuracy	Observations	Avg. Cost (s)
Random	93.23 ± 0.44	2.2 ± 0.18	3.92 ± 0.13
Heuristic (Star)	93.98 ± 0.31	2.2 ± 0.20	3.89 ± 0.16
Jeffrey's Divergence	96.48 ± 0.19	1.6 ± 0.21	3.01 ± 0.21
Loss Entropy	98.42 ± 0.21	1.4 ± 0.16	10.12 ± 0.34
Mutual Information	98.11 ± 0.36	1.4 ± 0.11	4.51 ± 0.27

Table 4.15: Kernel descriptors—Scenario where only the object's pose (yaw angle) has changed. The table shows the detection accuracy, number of observations needed as well as the total average computation cost in seconds.

Method	Accuracy	Observations	Avg. Cost (s)
Simple features			
No Change	98.3 ± 0.21	1.2 ± 0.19	0.79 ± 0.25
Pose Change	99.4 ± 0.12	1.3 ± 0.22	0.89 ± 0.32
Object Change	92.4 ± 0.15	1.9 ± 0.27	1.63 ± 0.46
Kernel descriptors			
No Change	100	1.4 ± 0.13	4.01 ± 0.48
Pose Change	100	1.6 ± 0.11	4.35 ± 0.29
Object Change	98.5 ± 0.15	2.1 ± 0.12	5.46 ± 0.38

Table 4.16: Quadcopter experiments—Feature selection and viewpoint selection are evaluated for the three different change detection scenarios using MI. The table shows the detection accuracy, number of observations needed as well as the total average computation cost in seconds.

and computation cost in seconds, evaluated for the simple features as well as the kernel descriptors. The quadcopter experiments confirm that detecting a complete change of an object with certainty is more difficult than detection in the other two cases. Furthermore, the quadcopter experiments show results comparable with the results of the simulation experiments evaluated on the two feature sets.

Discussion

The experimental results show that, given some prior information in form of a map, our framework can also be used to detect change in the environment. In our multi-view framework, on average, we can detect that no changes have occurred with an accuracy higher than 98% and a required number of equal or less than 1.4 observations. Detecting the change of the complete object, meaning the object was exchanged with another one, is

clearly a more difficult task. Results show that we can detect such cases with an average accuracy of 88–93%. Lastly, due to our ability of roughly estimating object orientations, we can also detect change in the yaw rotation of an object. For this case, we achieved average accuracies of 97% and more. As of now, our framework can only detect changes of objects in place, meaning translational changes would need to be detected separately, for instance by utilizing global position information.

4.6 Conclusion

We have presented an information-theoretic action selection framework for object recognition (and object change detection) which is capable of sequential training and classification, view planning as well as online feature selection. The recognition framework has successfully been tested in simulation and on a custom-built quadcopter robot equipped with an RGB-D camera.

Our online feature selection method has been shown to drastically reduce the computation time by computing only informative features used for inference during runtime. Online feature selection, compared to offline estimation, allows for much greater flexibility during training as well as execution. It is especially useful for robotic tasks like ours, since there decisions are made online, which stands in contrast to estimating the best performing set of features a priori during training phase. Furthermore, our experiments show that smart view planning, when we are uncertain about our belief state, can help to increase the recognition accuracy substantially.

Through extensive evaluations of various methods, we have shown the strengths and weaknesses in cost and accuracy of information-theoretic, heuristic as well as random strategies for feature and viewpoint selection. In general, one can say that information-theoretic viewpoint selection outperforms random selection in recognition accuracy, but increases computation cost. The selection of informative viewpoints is important to increase the accuracy and, at the same time, keep the number of observations minimal. In particular, this becomes apparent when using a simple heuristic like "motion along a star pattern". From an information point of view, moving to the opposite side yields a lot of new information. However, this information does not necessarily help us in distinguishing objects from each other in feature space—not every information is equally useful. Information-theoretic approaches allow for selecting the most informative actions in a principled way; they increase recognition accuracy and reduce operating cost.

Compared to state-of-the-art single-view object recognition methods, our multi-view object recognition approach, when used with simple features, achieves comparable results. Our approach is not primarily meant to replace single-view methods but rather to complement them. Put in other words, if a confident decision can be made about an object after only one observation, there will be no need for taking a second view. However, there will always be objects with inherent ambiguity, and it is in those cases in which the multi-view approach is brought to bear. Moreover, feature selection is useful regardless of the number

of views and the object in question, since it reduces the dimensionality of the feature space and decreases the overall computation cost.

Our framework can also utilize more complex features, such as the kernel descriptors from (Bo et al., 2011). Although these features are more expensive to compute, compared to the simple features, they result in an increase of recognition accuracy. An interesting trade-off can be seen here, the complex features have more expressiveness whereas the simple features are more cost-efficient to compute.

Chapter 5

Online Trajectory Optimization to Improve Object Recognition

There is growing interest in using robotic vehicles for inspection, to clear buildings, and to identify predefined targets. Because robots are replaceable, this interest is especially high when such tasks are dangerous. In order to clear the interior of a building, a robot must identify its surroundings while also avoiding collisions as it moves as efficiently as possible through the environment. Specifically, the robot should traverse the building while searching for and identifying specific objects, then reporting either their presence or absence. Assuming an optimal trajectory to fly through the building is computed offline, this could be computed from building blue print, the challenge would be to alter the initial trajectory online during flight so that the vehicle would avoid collisions and increase its chances of correctly identifying objects.

When the robot's sensors observe an object, different positions yield information of varying content and measurements of varying quality. This means some positions are much more useful than others in recognizing an object. For instance, if the camera is too far from the object, the image quality is poor; too close, and only a portion of the object is captured. Further, occlusions often prevent the sensors from observing the object from all angles, limiting the distinguishing features that are observable. The initial trajectory points of a robot's path might be optimal in terms of some cost function but unfavorable in terms of recognizing new objects in the environment. Consequently, we seek to optimize the initial trajectory in a way that significantly improves the robot's ability to acquire data that are useful in recognizing objects.

In this paper, we formulate an approach for optimizing the trajectory online that improves the ability to acquire information useful in recognizing objects while the vehicle approaches its goal location. Formulating the problem as a derivative-free stochastic optimization allows us to define arbitrary cost functions and thus improve upon the initial trajectory. The cost function we have developed alters the initial trajectory during execution to one that yields data for object recognition that are far superior to those observation data that would have been obtainable given the initial trajectory.

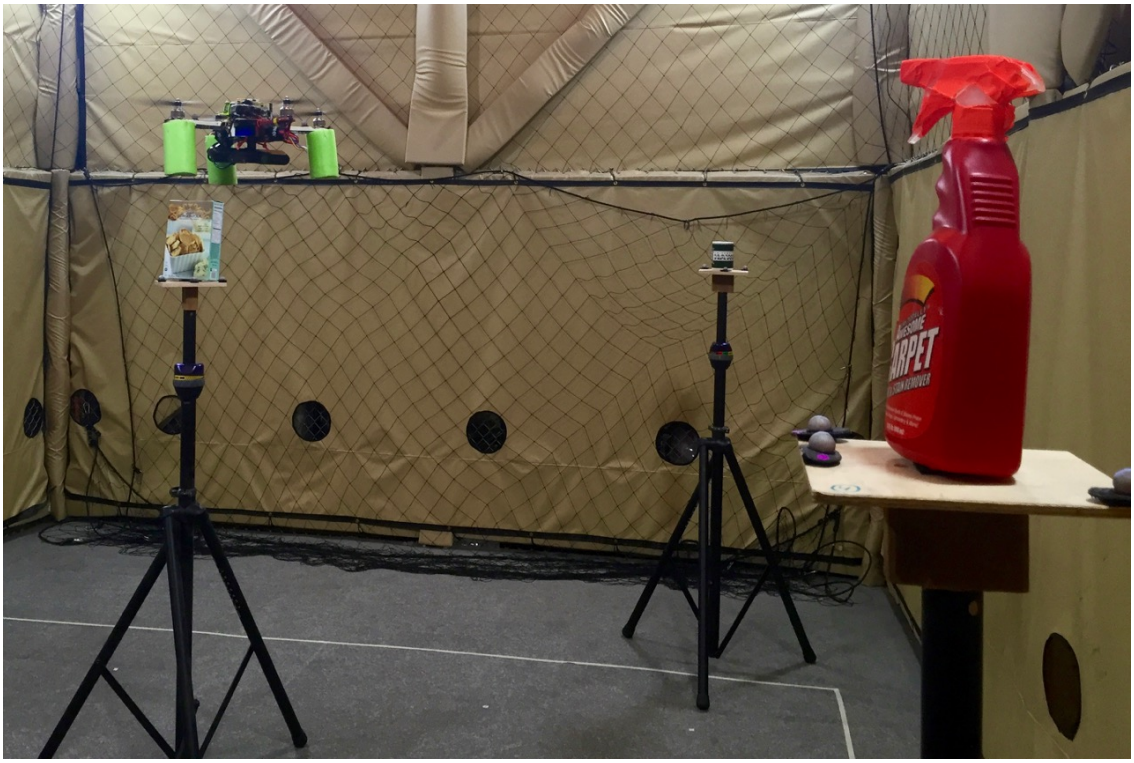


Figure 5.1: Our quadrotor platform taking an observation of a target object we want to recognize.

We demonstrate our approach both on a quadrotor platform in simulation and on a real platform Fig. 5.1. The quadrotor is given an initial trajectory that needs to be followed or altered online such that unknown objects are recognized while keeping the platform from colliding with it's environment. Our experiments show that our online optimization method greatly improves the accuracy with which objects are recognized, but more importantly reduces the posterior class distribution uncertainty greatly. In addition, we show that the number of observations needed to achieves these results are lower compared to following the initial trajectory.

Trajectory optimization methods can be found in many different robotic applications. A survey of current optimization techniques can be found in (Rao, 2014). Optimizing trajectories to fly aggressive maneuvers using a quadcopter is shown in (Richter, Bry, & Roy, 2013). Trajectory optimization for robotic manipulators is presented in (Kalakrishnan et al., 2011), (Ratliff, Zucker, Bagnell, & Srinivasa, 2009), (Park, Pan, & Manocha, 2012) and (Hauser & Ng-thow hing, 2010).

Target tracking for unmanned areal vehicles is presented in work by (Dille, 2013), (Theodorakopoulos, 2009), (Geiger, 2009) and (McEnteggart & Whidborne, 2012). Here, trajectory optimization is used to keep the target in the image frame and maximizes

surveillance time while observing the performance limits of the aircraft and accounting for external disturbances such as wind or target motion. In (Geiger, 2009) the authors present a neuronal network approach that removes the need to numerically compute the objective, constraint derivatives and removes the need for collocation, thus reducing the nonlinear programming problem size. A non real-time version is proposed in (McEnteggart & Whidborne, 2012) for a multi-objective trajectory optimization approach that finds environmentally efficient trajectories for commercial airplanes. The authors frame the problem as an optimal control problem and solve it by using a stochastic solver that solves for the states and controls of after the problem is discretized.

Active exploration (Kahn et al., 2015), (Kollar & Roy, 2008), (Charrow et al., 2015) uses trajectory optimization to gather more information which is unknown beforehand. In (Kahn et al., 2015) the trajectory of a robotic manipulator is optimized to explore the objects in the environment and find feasible grasps handles. Environment exploration and map building is presented in (Kollar & Roy, 2008) and (Charrow et al., 2015). In (Kollar & Roy, 2008) a trajectory is optimized and learned that lead to more accurate maps. In (Charrow et al., 2015) the authors develop a trajectory optimization technique that maximizes information collected from the environment to more efficiently build dense 3D maps while simultaneously satisfying the robot's motion constraints. The paper propose a gradient-based trajectory optimization that can refine a trajectory in the continuous control space using sequential quadratic programming.

As far as we know, there is no direct prior work of optimizing trajectories for object recognition. Our work is base on the stochastic trajectory optimization method presented in (Ratliff et al., 2009) and (Kalakrishnan et al., 2011). In there, the optimization and it's cost function is used to find feasible trajectories for a robotic manipulator. In our work, we formulate the optimization and cost function as an online optimization to optimize the trajectory of a quadrotor with the goal of improving the performance of object recognition.

5.1 Trajectory Optimization

Given an initial collision-free trajectory τ_{init} we want to find an optimized trajectory that improves recognition performance. In Fig. 5.2 we show a smoothed trajectory optimized to yield better measurements of the object we want to recognize. To this end, we treat the trajectory optimization problem as a stochastic optimization problem to search for a smooth trajectory that minimizes the costs corresponding to collision, constraints and improvement of classification. Specifically, we consider the initial trajectory to be of a fixed duration T, discretized into N waypoints. In other words, the discretized trajectory is composed of N configurations $q_1, \ldots, q_N$, where q_i is a trajectory waypoint at time $\frac{i-1}{N-1}T$. Furthermore, the trajectory is represented as a vector $Q \in \mathcal{R}^{\mathcal{D}*\mathcal{N}}$ and written as $Q = [q_1^T, q_2^T, \ldots, q_N^T]$. During the optimization, we assume that the start q_s and end q_e position of the trajectory are fixed.

At every waypoint, we update the trajectory along the future path of the vehicle. Since the vehicle can not observe its environment far into the future and to keep computation

time low, we do not optimize over the entire future trajectory every time. Instead, we only optimize over a fixed time or waypoint horizon. Let $K = \gamma$ be the number of waypoints within a given horizon γ. Then the trajectory we are optimizing over is given by $q_1, \dots, q_K$ with the start point $q_s = q_1$ and end point $q_e = q_K$.

Similar to previous work (Kalakrishnan et al., 2011) and (Ratliff et al., 2009) we formulate the optimization problem as following:

$$\min_{\tilde{Q}} \mathbb{E}\Big[\sum_{i=1}^{K} J(\tilde{Q}_i) + \frac{1}{2}\tilde{Q}^T R \tilde{Q}\Big], \tag{5.1}$$

with $\tilde{Q} = \mathcal{N}(Q, \Sigma)$ is a noisy parameter vector with mean Q and variance Σ. $J(\cdot)$ is an arbitrary state-dependent cost function, which we define in section 5.2. R is a positive semi-definite matrix representing the smoothness costs. Here, R is chosen such that $\tilde{Q}^T R \tilde{Q}$ represents the sum of squared accelerations along the trajectory. Let $\mathbf{A}$ be a finite differencing matrix

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 0 & & 0 & 0 & 0 \\ -2 & 1 & 0 & \dots & 0 & 0 & 0 \\ 1 & -2 & 1 & & 0 & 0 & 0 \\ & \vdots & & \ddots & & \vdots & \\ 0 & 0 & 0 & & 1 & -2 & 1 \\ 0 & 0 & 0 & \dots & 0 & 1 & -2 \\ 0 & 0 & 0 & & 0 & 0 & 1 \end{bmatrix},$$

when multiplied by the position vector Q produces acceleration $\ddot{Q} = AQ$, with $\ddot{Q}$ representing the second order derivative of the trajectory.

5.1.1 Stochastic Optimization

In general, solving the optimization in Eq. 5.1 is difficult. Previous work (Ratliff et al., 2009) has shown to solve the optimization using covariant functional gradient descent techniques. However, gradient descent techniques require the cost function to be smooth and differentiable, which is oftentimes difficult in practice. Instead, we follow the proposed method of (Kalakrishnan et al., 2011) and optimize Eq. 5.1 using a derivative-free stochastic optimization method. Allowing the optimization of an arbitrary cost functions, for which derivatives are not available, or are non-differentiable or non-smooth.

Taking the gradient of the expectation in Eq. 5.1, with respect to Q results in:

$$\delta \hat{Q}_G = \mathbb{E}\Big(\nabla_{\tilde{Q}}\Big[\sum_{i=1}^{N} J(\tilde{Q}_i)\Big]\Big) \tag{5.2}$$

As proposed in (Kalakrishnan et al., 2011) due to limitations of the gradient base optimization of non-smooth cost functions, we use a gradient of the form $\delta \hat{Q}_G = \int \epsilon \, \mathrm{d}\mathbf{P}$. This gradient is essentially the expectation of the noise ϵ in the parameter vector $\tilde{Q}$, under

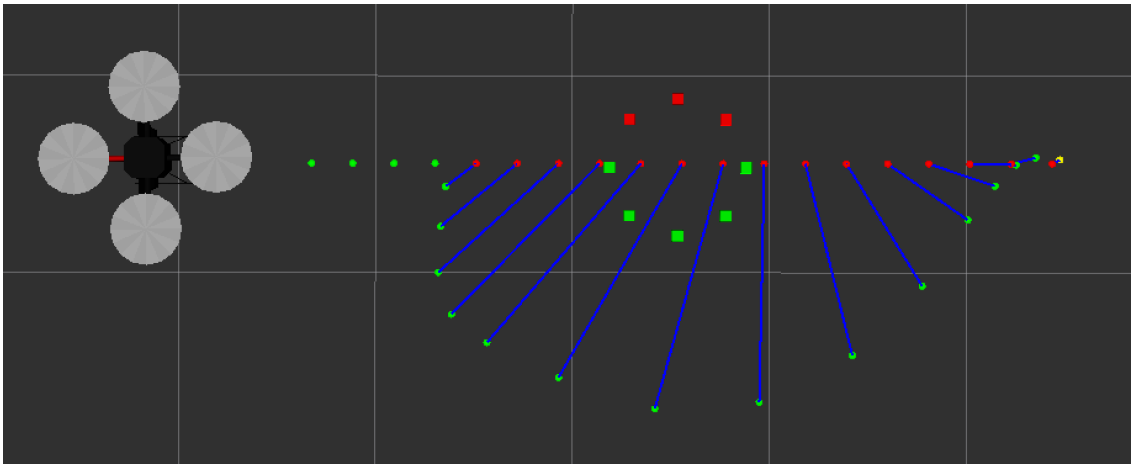


Figure 5.2: Smoothed optimized trajectory around an object we want to identify.

the probability metric $\mathbf{P} \propto \exp\Big(-\frac{1}{\lambda}J(\tilde{Q})\Big)$, with $J(\tilde{Q})\Big)$ being the trajectory cost. The stochastic gradient is then defined as follows:

$$\delta\hat{Q}_G = \int \exp\Big(-\frac{1}{\lambda}J(Q+\epsilon)\Big)\epsilon \ \mathrm{d}\epsilon \tag{5.3}$$

We estimate the gradient in Eq. 5.3 by sampling a finite number of U trajectories:

$$\delta\hat{Q}_G = \sum_{u=1}^{U} \mathbf{P}(Q+\epsilon_u)\epsilon \tag{5.4}$$

$$\mathbf{P}(Q+\epsilon_u) = \frac{\exp(-\frac{1}{\lambda}J(Q+\epsilon_u))}{\sum_{l=1}^{U}\exp(-\frac{1}{\lambda}J(Q+\epsilon_l))} \tag{5.5}$$

5.1.2 Trajectory Updates

In each iteration, we want to iteratively update the previously computed trajectory by generating U noisy sample trajectories. For this, we compute the trajectory cost per time-step $J(Q_u, t)$ as well as the probability of each trajectory per time-step $\mathbf{P}(Q_u, t)$. The trajectories are generated, by sampling the noise ϵ from a zero mean normal distribution, with $\Sigma_\epsilon = R^{-1}$ and $R^{-1} = (A^T A)^{-1}$. This ensures the exploration of the trajectory space and simultaneously keep the trajectory cost low.

Computing the probability per time-step requires us to evaluate the exponential term of Eq. 5.5. Here, λ is a parameter that regulates the sensitivity of the exponential cost.

$$e^{-\frac{1}{\lambda}J(Q_{u,t})} = e^{-h\frac{J(Q_{u,t})-\min J(Q_{u,t})}{\max J(Q_{u,t})-\min J(Q_{u,t})}}, \tag{5.6}$$

with $h = 10$ set to a constant.

Updating the trajectory Q is done using the probability-weighted convex combinations of noisy parameters for that specific time-step as shown in Eq. 5.4. However, to ensure a smooth updated trajectory, we smooth the gradients using a projection onto the basis vectors of R^{-1}. The trajectory update step thus looks as follows: $Q = Q + R^{-1}\delta\hat{Q}_G$. The iterative update rule of the trajectory guarantees that the average cost is non-increasing, if the sampling is assumed to be dense.

After updating K trajectory points we move the vehicle to the next waypoint and restart the optimization process to optimize the next K waypoints given new sensor measurements about the environment.

5.2 Cost Function

The cost function to optimize the trajectory consists of *obstacle cost* $c_o(q_i)$, *recognition cost* $c_r(q_i)$, *information cost* $c_i(q_i)$ and *center object cost* $c_v(q_i)$ and is computed as follows:

$$C(Q) = \sum_{i=0}^{K} c_o(q_i) + c_v(q_i) + \Big(c_r(q_i) * c_i(q_i)\Big). \tag{5.7}$$

The summation of the cost function over the horizon states is the sum over the obstacle cost, the object center cost and the product of the recognition and information cost. The product part of the equation becomes zero if we have collected sufficient information about the object. In other words, we have enough information if we are sufficient certain about the posterior estimate of the object class. In this case, we do not want to collect more data and with that stay close to the initial optimal trajectory, but still penalize trajectory points for being to close to an object the vehicle could collide with.

5.2.1 Obstacle Cost

To avoid collisions with obstacles we incorporate an obstacle cost that forces the vehicle away from the obstacle. Obstacles can be the objects we want to identify or other objects in the environment we have no knowledge off when the initial trajectory was created. Let all objects that are closer as τ of the vehicle to the trajectory point be defined as O_c with $c \in \{1, \ldots, M\}$ and their position be defined as $\vec{o}_c$.

The obstacle cost is computed by approximating the vehicle with a sphere b and is computed as follows:

$$c_o(q_i) = \sum_{c=1}^{M} \max(\epsilon + r_b - dist(\vec{x}_0, \vec{o}_c), 0)\|\dot{\vec{x}}_b\|, \tag{5.8}$$

with $dist(\cdot,\cdot)$ being the euclidean distance between two points, r_b begin the radius of the sphere, $\vec{x}_o$ being the 3D position of the sphere b and ϵ begin a safety margin. To prevent

the vehicle of traversing high cost region with high speed, we multiply the cost by the magnitude of the workspace velocity $\|\dot{\vec{x}}_b\|$ of the sphere b.

The obstacle cost $c_o(q_i)$ is zero if all obstacles are at least ϵ away of the vehicle sphere b.

5.2.2 Center Object in Camera View

The center object cost keeps the target object in the center of the image plane. The cost is defined such that it has it's minimum value when the target object is in the center of the image plane and reaches its maximum when the object is out of the camera frame. The center object cost function is calculated by transforming the target's world coordinates into image plane coordinates, $\vec{v} = [v_x, v_y, 1]$, using a perspective plane transformation. We express the center object cot as with the following formulation:

$$c_v(q_i) = \min(v_x^2 + v_y^2, 1) = \min(\vec{v} \cdot \vec{v}, 1). \tag{5.9}$$

If no object is in close proximity of the vehicle we set $\vec{v}$ such that the next waypoint of the trajectory is in the center of the image plane. This way, the camera is directed into the direction the vehicle moves and can spot potential objects or targets.

5.2.3 Recognition Cost

To increase recognition accuracy and improve the certainty of the posterior class estimate, we incorporate a recognition cost into our cost function. Intuitively, we want to have the vehicle keep the target object O_r, with position o_r at a certain distance such that the object is completely in frame of the camera. This is due to the fact that partially observed objects can lead to wrong object class estimates. Furthermore, we want to steer the vehicle to new observation directions o_v with $v \in 1, \ldots, V$ to collect new information about the object in question. The observation direction is a point around the object to which the robot is maneuvered as shown in Fig. 5.2 as green and red squares. We choose the observation direction closest to the robot as the one to be attracted to. Once we have taken an observation from this position, we treat the view of the object as observed and want to steer the vehicle to the next observation point along the path of the vehicle. This means we select the next closest viewpoint along future waypoints as the next observation position o_v.

The object recognition cost function is defined as follows:

$$c_r(q_i) = dist(o_r, q_i) + PLdist((o_r, o_v), q_i) \tag{5.10}$$

with $PLdist((\cdot,\cdot), q_i)$ as the distance between the waypoint q_i and the line passed through the object target o_r and the observation point o_v. The distance attracts the vehicle closer to the observation point, while the point to line distance attracts it to the right observation angle.

5.2.4 Information Cost

We only want to collect as much information about the target object as necessary to achieve high probability identifying the object correctly. Once enough information is collected, we want to proceed towards the goal or focus on an object that is in close vicinity of the vehicle. A measure of uncertainty is defined in the *Shanon entropy*, which measures the uncertainty in a random variable. The entropy is zero if the outcome of the classification is unambiguous and it reaches its maximum if all object classes are equally likely. The entropy of the posterior object classification distribution is defined as $H(o_t|f_{1:t}) = -\sum P(o^i|f_{1:t}) \log P(o^i|f_{1:t})$. We define the information cost c_i such that it becomes zero if we are sufficiently certain about the class estimate given by the MAP estimate of the posterior distribution. Therefore, we define the information cost as:

$$c_i = \begin{cases} 0, & \text{if } H(o_t|f_{1:t}) < \eta \\ 1, & \text{otherwise} \end{cases} \tag{5.11}$$

Here, if the entropy $H(o_t|f_{1:t})$ becomes smaller then η we set $c_i = 0$, indicating we are certain about the object class of the target object.

5.3 Object Recognition

Given a static scene and an object in question we want to correctly recognize the object. The recognition process matches an observation of the target object with the objects stored in a database and computes the belief states of all possible objects. Due to noise, ambiguity and occlusion the recognition process does not always lead to certain matchings, which in turn results in multiple likely object candidates. To this end, we seek the minimal number of observations that accurately identifies the object in question.

During object observation, the sensor captures the raw data, such as color images and 3D point clouds, which is used to compute the features. The sensor measurements of a new observation are parametrized by the feature vector $\vec{f} = [f^1, \ldots, f^N]$, which includes N different feature types. The observation uncertainty is captured in our observation model, defined as the conditional probability density function $p(\vec{f}_z|o_z)$, with recognition state o_z and feature vector $\vec{f}_z$ captured at time z. In the recognition state estimation, we use a *Bayesian framework* to infer about the current object class o_z from the measurements. The parameters are assumed to be obtained from a physical object model and to be trained offline.

5.3.1 Sequential Bayesian Recognition

With every new observation of a feature $\vec{f}_{z+1}$, we want to efficiently integrate the new observation with results from prior observations, and thus update our current belief about the object class. We formulate the recognition process over the object classes as a Bayesian network in form of a *Hidden Markov Model* (HMM). A Bayesian formulation has distinct

advantages, with the main advantage being that Bayesian methods model all sources of uncertainty in form of random variables. Furthermore, HMMs have efficient formulations to sequentially update the posterior distribution, making it ideal for integrating new observations, such as in our case in the form of a new feature $\vec{f}_{z+1}$. Compared to a naive Bayes update, which is often used in existing work (Denzler & Brown, 2002; Paletta & Pinz, 2000), the HMM formulation additionally gives us the possibility to explicitly formulate the *state transitions*.

The joint probability distribution is defined as

$$p(o_{1:Z}, \vec{f}_{1:Z}) = p(o_1) \left[\prod_{z=2}^{Z} p(o_z|o_{z-1}) \right] \prod_{z=1}^{Z} p(\vec{f}_t z|o_z) \,. \tag{5.12}$$

The observation sequence is $S = \{o_1, \ldots, o_z\}$, the hidden state of our HMM is defined as recognition state o_z with $o_{1:Z} = \{o_1, \ldots, o_z, \ldots, o_Z\}$ and the observations are given as observed features $\vec{f}_z$ with $\vec{f}_{1:Z} = \{\vec{f}_1, \ldots, \vec{f}_z, \ldots, \vec{f}_Z\}$. The state of the hidden process satisfies the Markov property, that is, given the value of o_{z-1}, the current state o_z is independent of all the states prior to $z-1$.

The *observation model* $p(\vec{f}_z|o_z)$ represents the likelihood that feature $\vec{f}_z$ is generated by recognition state o_z. The transition probability is defined as $p(o_z|o_{z-1})$, which means that the transition to the next recognition state is dependent on the previous recognition state (i.e., the previous object class). The posterior marginals $p(o_z|\vec{f}_{1:Z})$ of all hidden state variables, given a sequence of observations, can be efficiently computed using the *forward-backward algorithm*. The algorithm computes a smoothed estimate given all evidence in two passes, with the first pass going forward in time and the second pass going backwards in time:

$$\begin{aligned} p(o_z|\vec{f}_{1:Z}) &= p(o_z|\vec{f}_{1:z}, \vec{f}_{z+1:Z}) \\ &\propto p(\vec{f}_{t+1:Z}|o_z)\; p(o_z|\vec{f}_{1:z}) \,. \end{aligned} \tag{5.13}$$

In the first pass, we compute the forward probabilities, which are the probabilities for all $z \in \{1, \ldots, Z\}$ ending up in any particular state given the first z observations in the sequence $p(o_z|\vec{f}_{1:z})$. In the second step, we compute the probabilities $p(\vec{f}_{z+1:Z}|o_z)$ of making the remaining observations given any starting point z.

5.3.2 Observation Model

For recognition, we learn a function $g : \mathbb{R}^{|\vec{f}|} \to \mathbb{N}$ that maps the extracted object features $\vec{f}$ to a unique object identifier $i \in \{1, \ldots, K\}$, each pointing to a distinct object model $o^i = c^k$, with $k \in \{1, \ldots, K\}$ and K denotes the total number of object classes c^k. The function is learned in a supervised fashion, with all possible features $\vec{f}$ and the unique identifier i available as the value pair $(\vec{f}, i)$ at model training phase.

The observation model is defined as a Gaussian generative model of the form $P(f|o^i) = \prod_{m=1}^{M} \mathcal{N}(f|\mu_{o^i,m}, \sigma_{o^i,m})$, with M is the number of independent normal distributions to model a feature f. Each feature f is represented as a feature histogram with M bins,

and the mean $\mu_{o^i,m}$ and variance $\sigma_{o^i,m}$ for each bin m is computed over the training data of an object with recognition state o^i. We use M independent normal distributions since a multivariate normal distribution, with full covariance matrix, would become high-dimensional for high-dimensional features. High-dimensional models are in general poorly separable as is well understood under the *curse of dimensionality*.

We can now compute the likelihood $p(\vec{f}|o^i)$ of generating the model o^i from feature vector $\vec{f}$ as

$$P(\vec{f}|o^i, \vec{\psi}) = \sum_{n=1}^{N} \psi^n \sum_{m=1}^{M} \mathcal{N}(f^n|\mu_{o^i,m}, \sigma_{o^i,m}), \tag{5.14}$$

with ψ^n denoting a scaling factor to scale the different feature distributions. The Gaussian representation of the features is compact and fast to train, since it only requires the computation of the Gaussian feature parameters $\mu_{o^i,m}$ and $\sigma_{o^i,m}$. Furthermore, sequential learning as well as sequential updates of mean and variance are possible, which has the potential of adaptively adding new observations to our trained features.

Instead of learning just one model per object class, we learn multiple models. This has the advantage that the model trained for each object has a smaller variance if trained only for a partial view of the object. This becomes intuitively clear since an object looks in general very different for different viewing directions. And this again is reflected in the variance of the model.

5.3.3 HMM State Transition

We define the state transition probability $p(o_z|o_{z-1})$ as the probability of a transition from recognition state o_{z-1} to state o_z. This means, how likely is the transition, if at time $z-1$ we believe to be in recognition state $o_{z-1} = c^k$ (i.e., a believed target object with object class c^k), then move the vehicle to a new waypoint and acquire a new observation and feature $\vec{f}_z$, and finally believe to be in state $o_z = c^k$ at time z.

In order to determine the optimal state transition probabilities, we uses the *Baum-Welch algorithm*, which uses expectation maximization to find the maximum likelihood estimate of the state transition parameters given a set of observed data.

5.4 Experiments

In this section, we present the evaluation results of our framework on a quadrotor platform both in simulation and on a real platform. The quadrotor is given an initial trajectory, which is to be followed or altered, such that the vehicle can recognize objects along its path. At the beginning of the mission, position, identity and number of objects in the environment are unknown and need to be discovered as the robot makes progress along the given initial trajectory. As the robot moves through the environment, along the waypoints, the quadrotor is able to detect objects of interest up to a distance of 3.0m. As the robot gets closer to the current target object $\leqslant 2.0$m, data is collected and used as input to the object recognition framework.

The object recognition framework is trained on a total of 326 objects. The training set is comprised of a publicly available RGB-D dataset introduced in (Lai et al., 2011) and objects recorded by us. The RGB-D dataset consists of a color image, a 3D point cloud and the orientation of the object, for a total of 300 everyday objects. However, since we do not have any of the objects or models we can use for our simulation or real world experiments, we created the same data for 26 additional objects. For real world experiment we record data from real object while in simulation we create a dataset from virtual models. For each object, the data is recorded from directions all around the object, including three different viewing angles of 30°, 45° and 60° per direction. We train our object models from $\vec{f}$ using data captured from viewpoints at 30° and 60° in each direction, and test them against features computed from data that is recorded from viewpoints at 45°.

In our experiments we only select objects from our dataset, however the inference is done over the entire 326 object dataset, to increase overall difficulty. The overall dataset is challenging, since the objects in the dataset look very similar, especially when they belong to the same object category. Furthermore, the objects in the dataset exhibit large changes in illumination due to different viewing angles from which the objects were captured, which may make classification difficult even across different categories.

The feature vector $\vec{f}$, which is used throughout our framework, is comprised of three independent features: object bounding box, color histogram and SIFT (Lowe, 2004). Each feature represents a different object property: size, color and a scale-invariant image descriptor. In accordance with eq. 5.14, each single feature or component of a feature histogram, respectively, is expressed as independent normal distribution. The scaling factor ψ^n is chosen empirical and set to $\{Color, Bbox, SIFT\} = [1.0, 0.02, 0.2]$. For each object instance we train in total eight different models which divides the training set into 45 degree viewing intervals. All reported accuracies of our recognition results are expressed as *recall* and stem from the MAP estimated of the posterior distribution.

In this work the features and recognition framework is rather simple, compared to a state-of-the-art system. In a single-view recognition scenario the recognition performance of our system in a controlled environment is rather poor performing with an accuracy of only 82.35%. The advantage of the system however, is that the features and class

predictions can be computed extremely fast. This makes it possible to run the entire system in real time. Nevertheless, the recognition framework actually performs quite suitably in a multi-view scenario, demonstrating that even simple features are capable of performing well when additional information is collected and utilized as is shown in (Potthast et al., 2015). That being the case, the recognition system used herein is not novel or state of the art. In fact, any other Bayesian recognition framework could be substituted for the one herein employed.

We test the recognition performance with two different path following methods. In the base case, the robot is just blindly following the initial waypoints, collecting data as it progresses through the environment. The second method is our online trajectory optimization method, as the robot moves through the environment, the path is optimized to minimize the cost computed in eq. 5.7. The trajectory and with that the waypoints are represented in cartesian space as well as a yaw angle for orientation and is defined as $q_i = [x, y, z, \theta]$. To keep computation time reasonable and still be able to update the trajectory fast, we choose the waypoint horizon $\gamma = 10$. To penalize a potential collision with an object, we define the parameters of the obstacle cost with $\tau = 2.0$m as objects that potentially could collide with the quadrotor, the safety margin $\epsilon = 0.5$m and the robots sphere $r_b = 0.5$m. The information cost entropy parameter which defines the cut-off as when to stop collecting more data is set to $\eta = 0.1$. All experiments are conducted over 10 trials, each time, the set of objects is exchanged randomly and the robot placed at the beginning of the initial trajectory.

5.4.1 Simulation results

We test the performance of our trajectory optimization method in three different simulation scenarios, varying in the initial trajectory and the number of objects we need to identify. We show the three scenarios with their initial trajectories and object placement in Fig. 5.3, 5.5 and 5.7 and the optimized trajectories in Fig. 5.4, 5.6 and 5.8. In both versions, the non-optimized and optimized version, the waypoints that the quadrotor has followed are shown in green dots. Objects that the robot needs to identify are circled with square boxes. The square boxes represent the observation directions o_v of the target object, a green box represents the direction from which we have taken a measurement, a red box is a direction from which we have not collected any information yet. For each observation direction, we only capture one observation.

The quantitative results of the simulation experiments are shown in Table 5.1 and Table 5.2 . We can see that the time, on average, the vehicle needs to traverse the trajectory in the non-optimized version is shorter compared to the optimized version. This is due to the time the online trajectory optimization algorithm takes to compute and update the trajectory. The optimized version on the otherhand enables the vehicle to collect better information about the target object as we can see in the accuracy of recognizing the object correctly. On average the optimized trajectory achieves much better accuracy, but more importantly we can see that the average entropy of the posterior estimation is much lower. A lower entropy estimate means we are more certain of the class estimate of the target

	#Waypoints	#Objects	Avg. Time(sec)
(1) Initial Trajectory	45	5	99.8 ± 2.3
(1) Optimized Trajectory	45	5	115.6 ± 4.0
(2) Initial Trajectory	45	7	94.7 ± 4.0
(2) Optimized Trajectory	45	7	115.5 ± 3.2
(3) Initial Trajectory	56	10	104.9 ± 2.2
(3) Optimized Trajectory	56	10	135.1 ± 2.8

Table 5.1: Simulation results – The table shows the average performance by following the initial trajectory or optimized trajectory. The three scenarios are dissimilar in number of total waypoints and object to be recognized. All results are averaged over 10 different trials, with the objects we want to recognize chosen at random.

	Avg. Observations	Avg. Accuracy	Avg. Entropy
(1) Initial Trajectory	22	72.0% ± 13.3	0.26 ± 0.03
(1) Optimized Trajectory	14.7 ± 2.6	92.0% ± 9.8	0.12 ± 0.05
(2) Initial Trajectory	22	61.43% ± 11.16	0.28 ± 0.05
(2) Optimized Trajectory	18.8 ± 0.9	94.29 ± 7.0	0.10 ± 0.03
(3) Initial Trajectory	28	69% ± 15.13	0.27 ± 0.02
(3) Optimized Trajectory	22 ± 1.7	93.0 ± 7.81	0.11 ± 0.02

Table 5.2: Simulation results – The table shows the average performance by following the initial trajectory or optimized trajectory. We show the needed average observations, average accuracy and average total entropy of the objects to be recognized.

objects. A higher entropy of the posterior class distribution is the result of partial or poor quality target object observations. This can happen if the vehicle is too far away or if the vehicle is too close and the object does not fit completely into the image frame. Additionally, we can see that less observations are needed which can also be attributed to the fact of better observations, but also due to the fact that we stop collecting data once we have enough certainty about the target object.

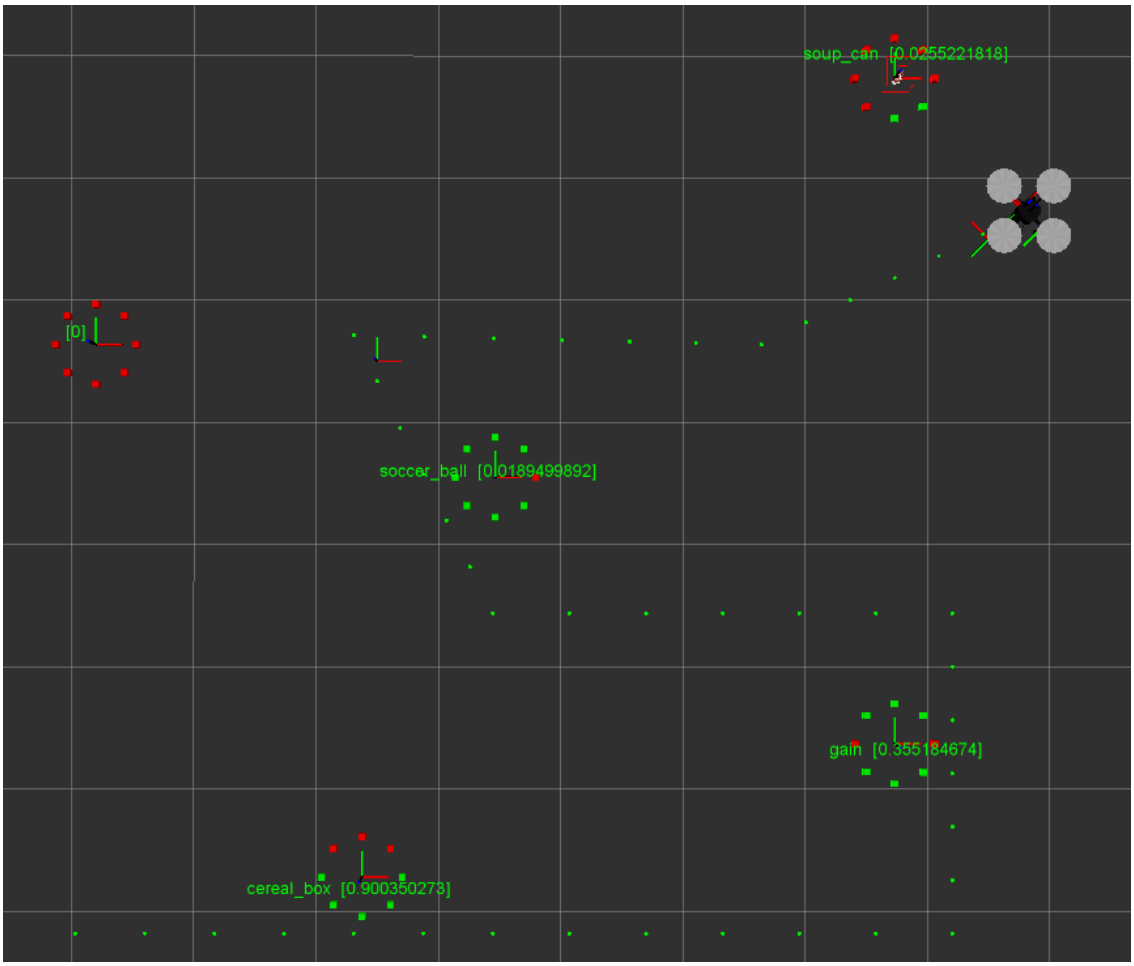


Figure 5.3: Simulation Experiment 1 – In this experiment the vehicle follows the initial trajectory shown as green dots. The vehicles goal is to follow the trajectory and recognize 5 objects depicted as circles with green/red boxes. The green/red boxes show the possible observation angles, once an observation has been taken from a certain angle the box is marked as red and no other observation is taken with similar angle.

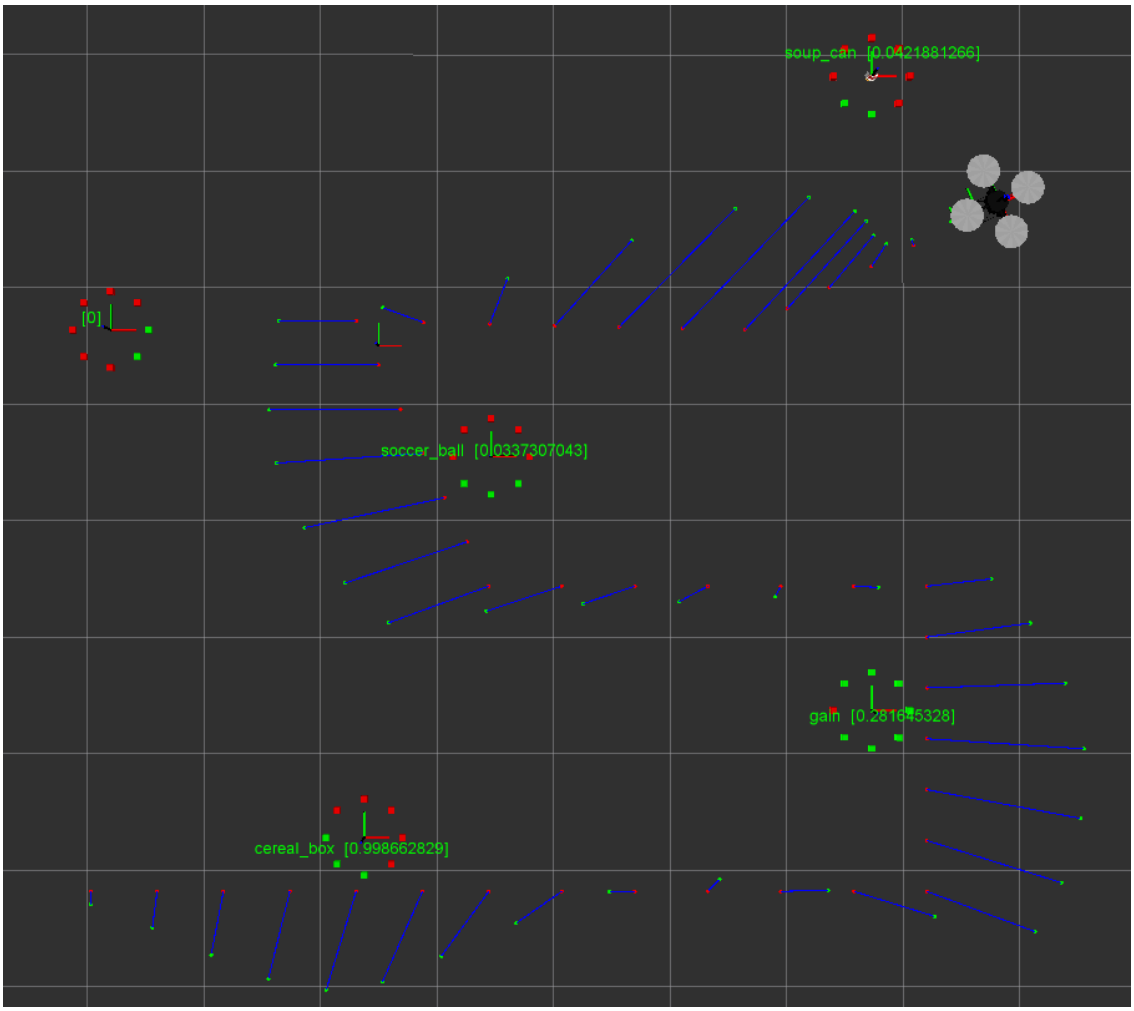


Figure 5.4: Simulation Experiment 1 – In this experiment the vehicle follows the online optimized trajectory. The red dots show the initial trajectory while the green dots show the optimized trajectory given the initial waypoints. The new trajectory enables the vehicle to more reliable recognize the 5 objects depicted as circles with green/red boxes. The green/red boxes show the possible observation angles, once an observation has been taken from a certain angle the box is marked as red and no other observation is taken with similar angle.

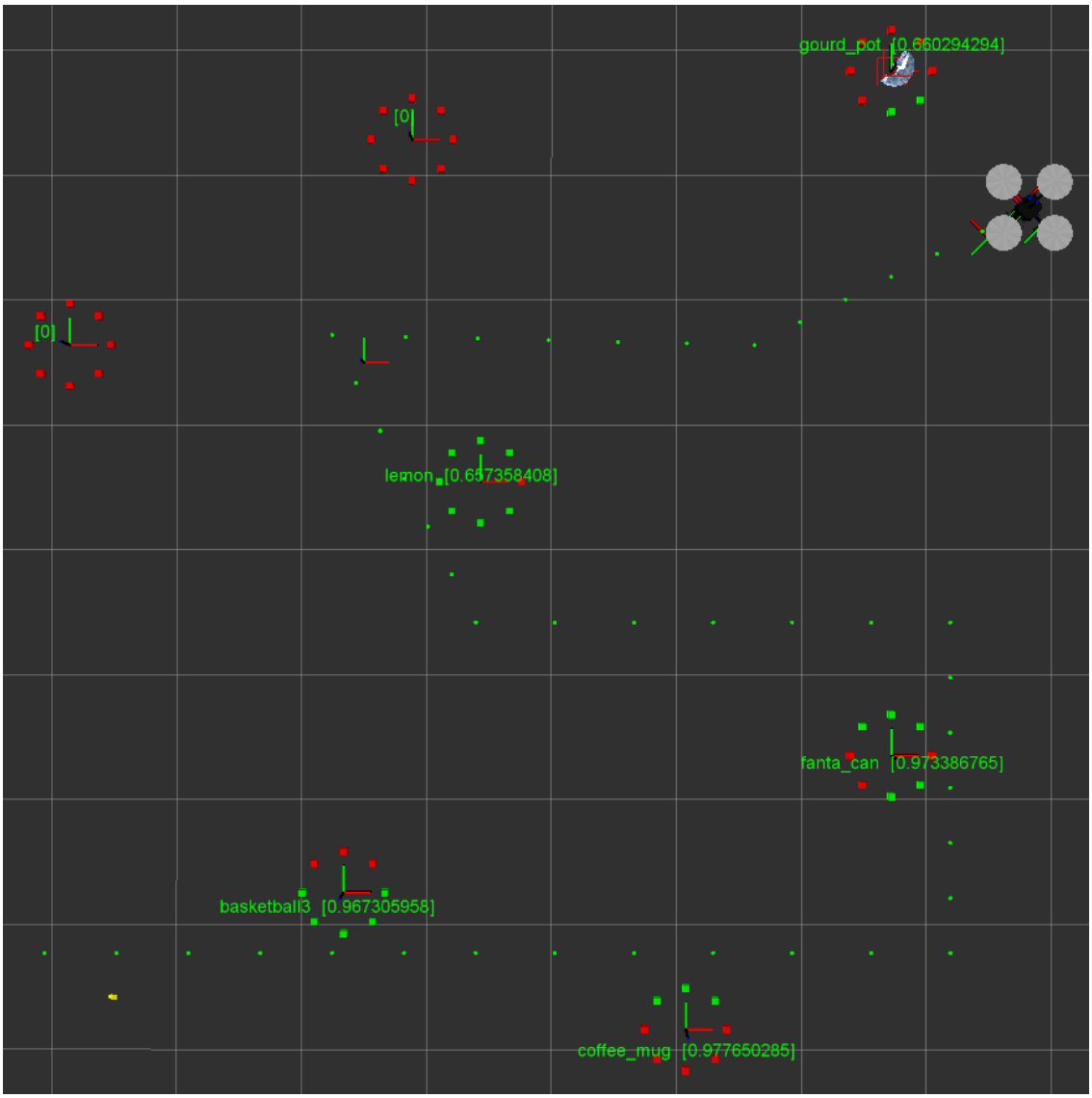


Figure 5.5: Simulation Experiment 2 – In this experiment the vehicle follows the initial trajectory shown as green dots. The vehicles goal is to follow the trajectory and recognize 7 objects depicted as circles with green/red boxes. The green/red boxes show the possible observation angles, once an observation has been taken from a certain angle the box is marked as red and no other observation is taken with similar angle.

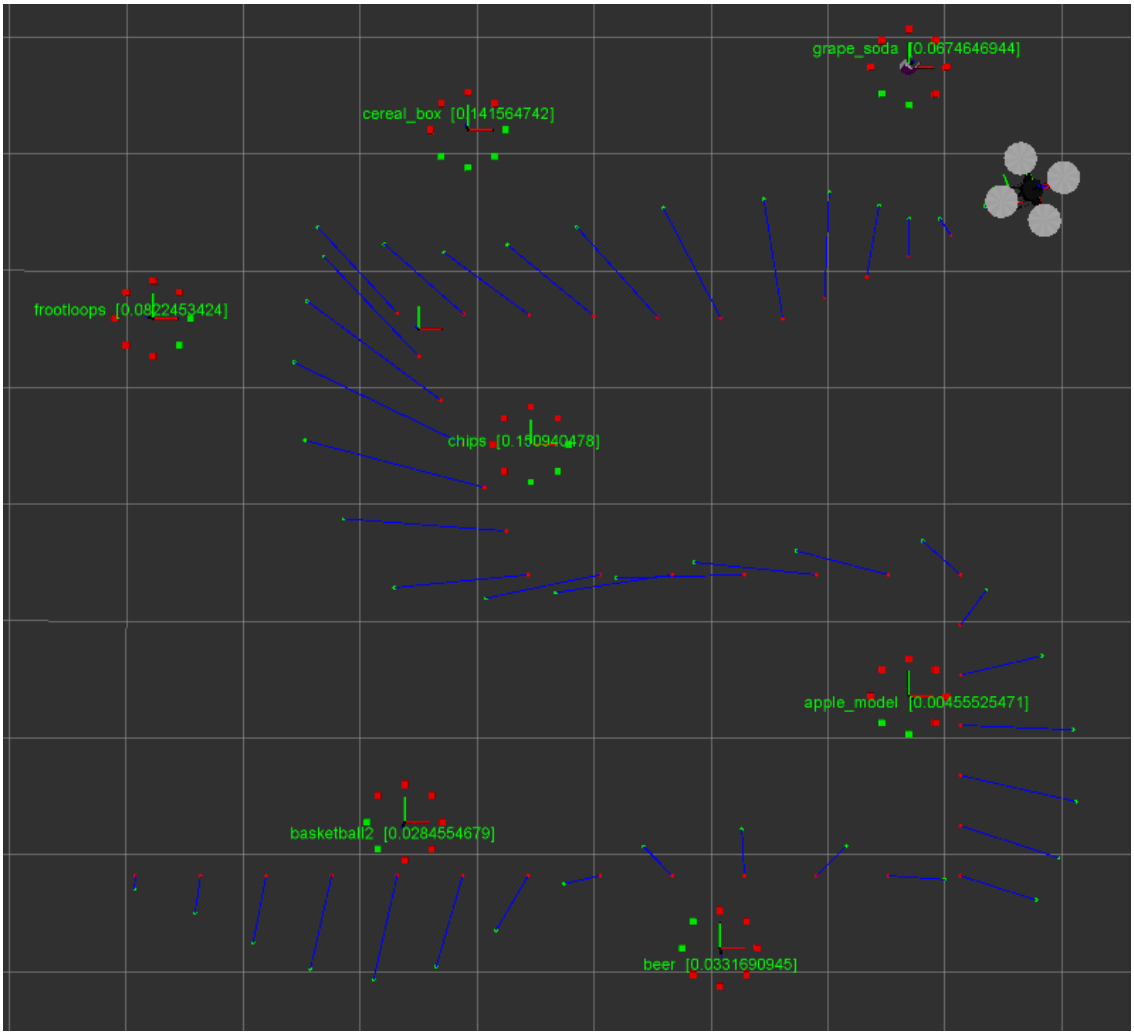


Figure 5.6: Simulation Experiment 2 – In this experiment the vehicle follows the online optimized trajectory. The red dots show the initial trajectory while the green dots show the optimized trajectory given the initial waypoints. The new trajectory enables the vehicle to more reliable recognize the 7 objects depicted as circles with green/red boxes. The green/red boxes show the possible observation angles, once an observation has been taken from a certain angle the box is marked as red and no other observation is taken with similar angle.

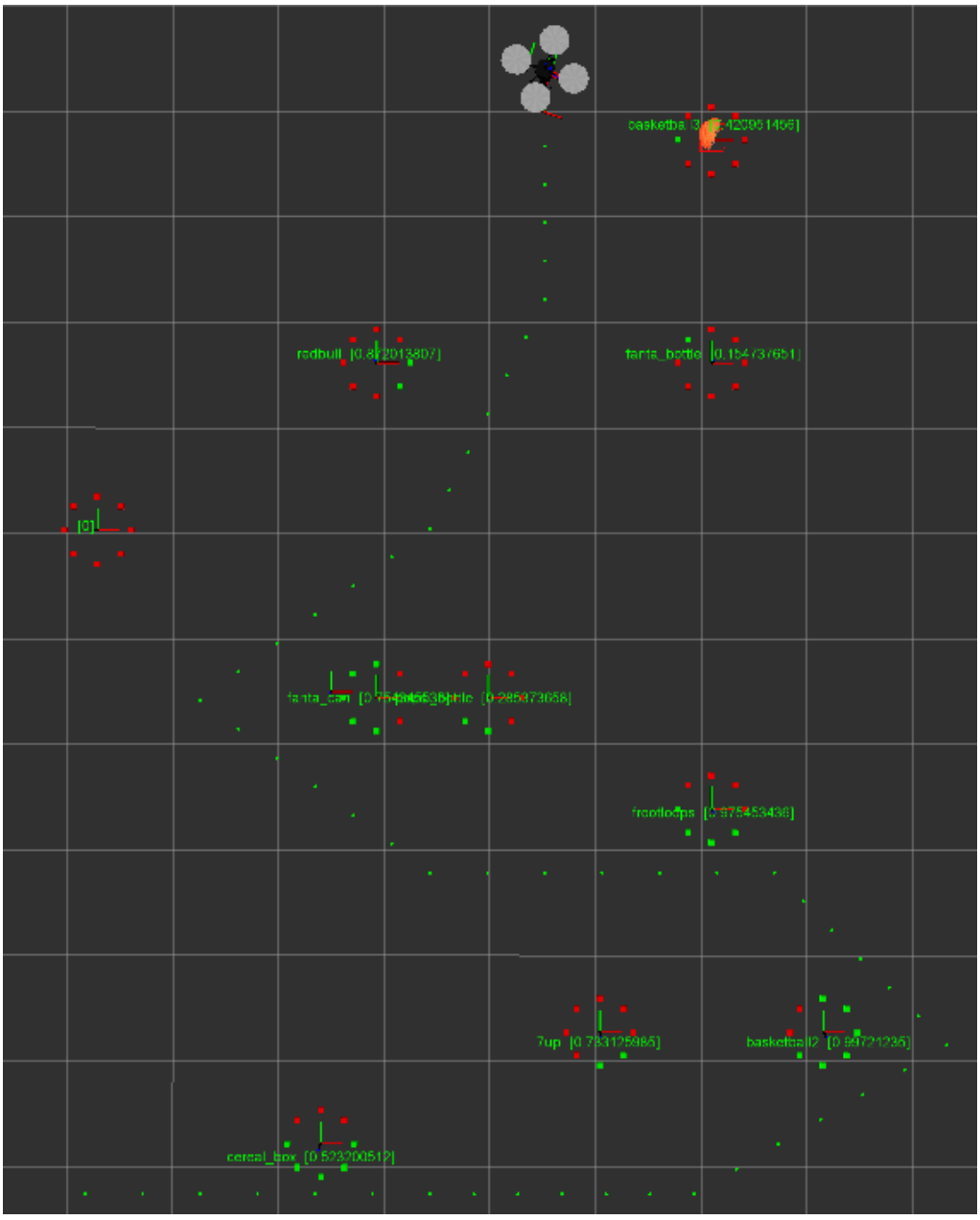


Figure 5.7: Simulation Experiment 3 – In this experiment the vehicle follows the initial trajectory shown as green dots. The vehicles goal is to follow the trajectory and recognize 10 objects depicted as circles with green/red boxes. The green/red boxes show the possible observation angles, once an observation has been taken from a certain angle the box is marked as red and no other observation is taken with similar angle.

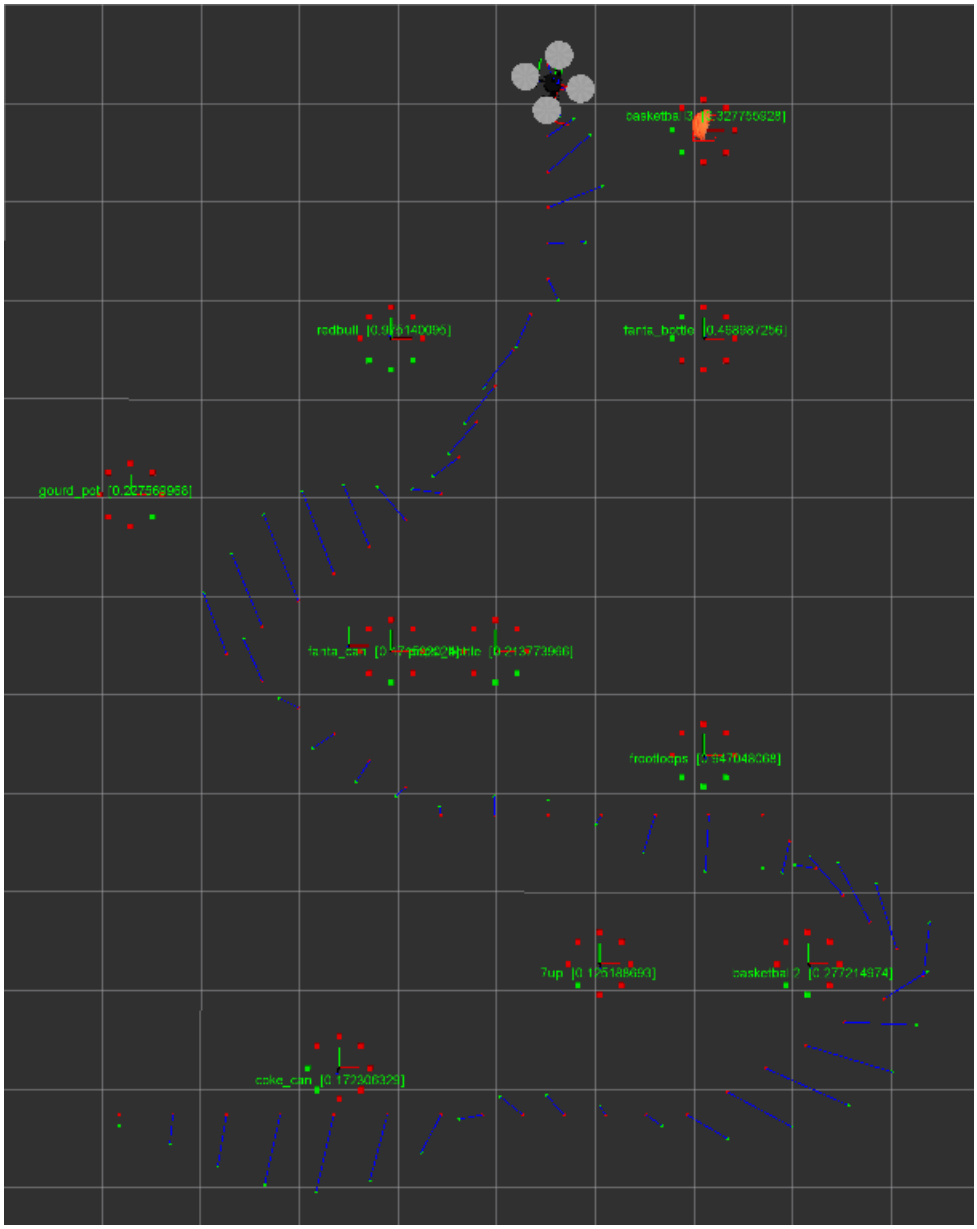


Figure 5.8: Simulation Experiment 3 – In this experiment the vehicle follows the online optimized trajectory. The red dots show the initial trajectory while the green dots show the optimized trajectory given the initial waypoints. The new trajectory enables the vehicle to more reliable recognize the 10 objects depicted as circles with green/red boxes. The green/red boxes show the possible observation angles, once an observation has been taken from a certain angle the box is marked as red and no other observation is taken with similar angle.

5.4.2 Quadrotor results

In order to show the real world performance of our online trajectory optimization method, we have conducted experiments on our custom-build quadrotor platform. As shown in Fig. 5.1, the quadrotor is equipped with an ASUS Xtion RGB-D camera for data acquisition and can fly autonomously along the waypoints of the trajectory. The experiment is setup identically to the simulation experiment, with the quadcopter optimizing the trajectory online, following the updated waypoints and collecting data of target objects along the way. The results of the real world experiment are shown in Table 5.3. Similar to the simulation results, we can see a similar performance of the optimized trajectory compared to the non-optimized trajectory. The smooth optimized trajectory achieves a higher average recognition accuracy with a lower average uncertainty of the MAP estimate. Furthermore, the optimization results in a lower average number of observations taken.

	#Waypoints	#Objects	Avg. Time(sec)
Initial Trajectory	44	4	58.8 ± 1.7
Optimized Trajectory	44	4	81.7 ± 1.35
	Avg. Observations	Avg. Accuracy	Avg. Entropy
Initial Trajectory	15	74.29 % ± 17.04	0.27 ± 0.02
Optimized Trajectory	10.8 ± 1.8	90% ± 12.25	0.13 ± 0.01

Table 5.3: Quadrotor results – Average performance with random objects placed into the scene over 10 different trials

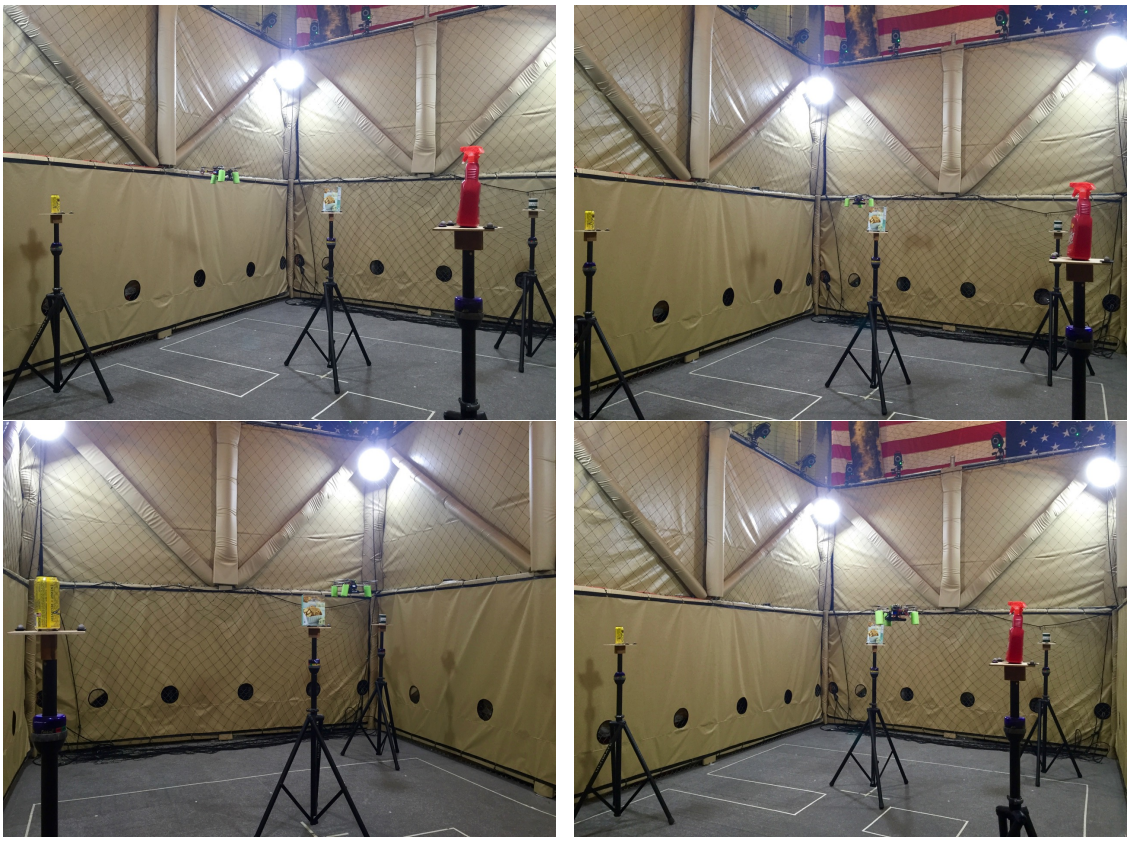


Figure 5.9: From top left to bottom right we show in order how the quadrotor captures information about the target objects along the optimized trajectory.

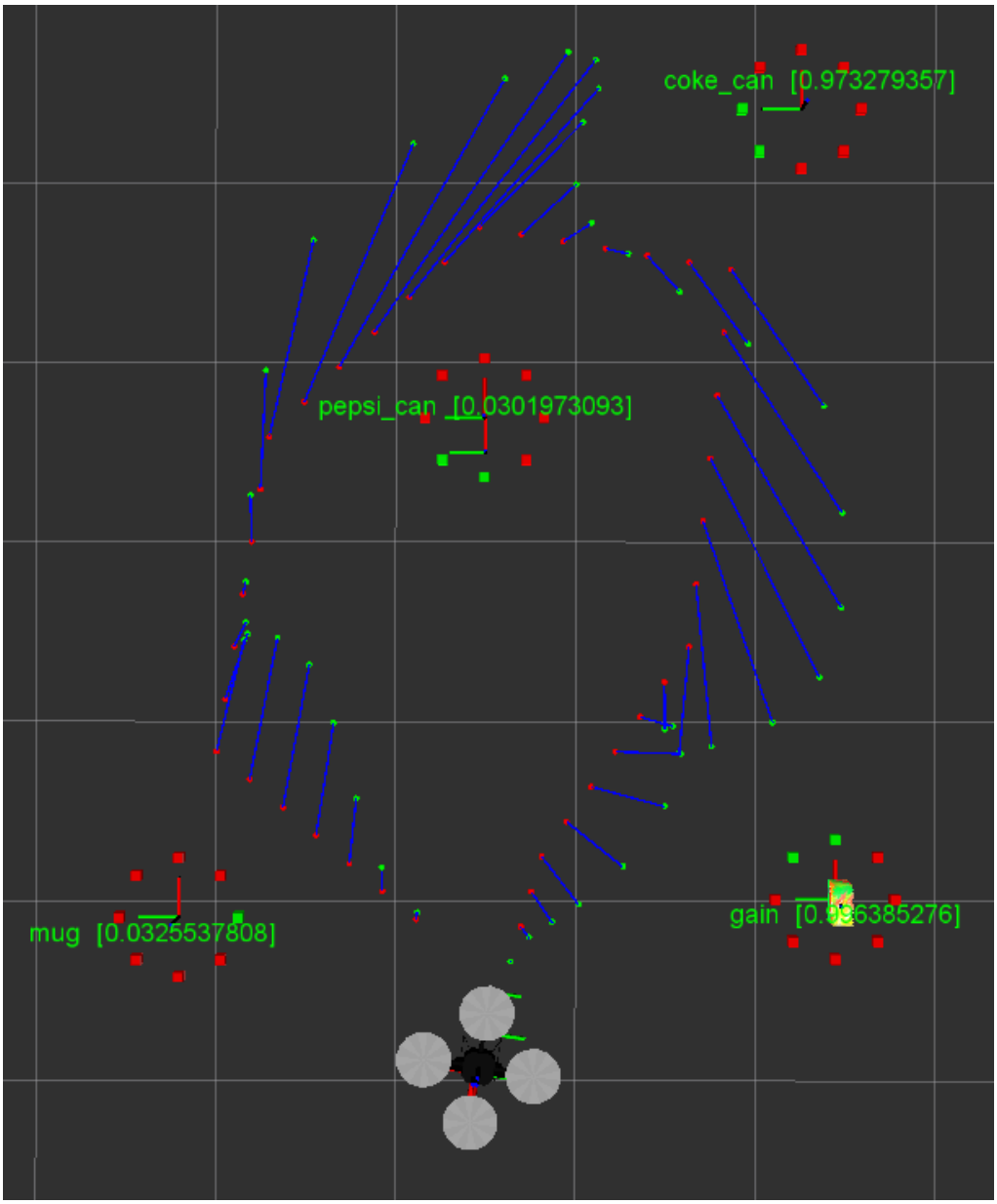


Figure 5.10: Quadrotor results – Optimized trajectory of the quadrotor, 4 objects are being recognized.

5.5 Conclusion

In this work, we have presented an online trajectory optimization approach that optimizes waypoints towards improving recognition of objects. The formulated stochastic optimization approach allows us to change the trajectory in real-time such that observations taken of a target object result in more accurate recognition estimates of the target object. We achieve this by preventing inference with partial observations of the target object and in addition by taking the uncertainty of the posterior distribution into account. Considering the entropy of the posterior class distribution allows us to determine whether we need to collect more information about the current target object or if the information collected yields enough information to make a class prediction of high certainty. In numerous experiments, simulation as well as on a real robot, we have shown the usefulness of our approach. We have shown that the optimized trajectories not only achieve, on average, a higher recognition accuracy, but more importantly that the posterior class distribution is less uncertain. This results in more certainty of the class prediction given by the MAP estimate of the posterior distribution. Currently, all computation, trajectory optimization as well as object recognition, is run on an off-board computer, a future plan is to run everything onboard the quadrotor.

Chapter 6

Conclusions

In this thesis we have presented information theoretical frameworks for data acquisition and scene understanding. In extensive experiments we have shown that adaptive action selection is a useful tool for robots to understand the surrounding they operate in. In this chapter, we summarize the contributions made in this thesis.

First, in Chapter 3 we presented an information gain-based variant of the next best view problem for occluded environment. Our proposed method utilizes a belief model of the unobserved space to estimate the expected information gain of each possible viewpoint. More precise, this belief model allows a more precise estimation of the visibility of occluded space and with that a more accurate prediction of the potential information gain of new viewing positions. We presented experimental evaluations on a robotic platform for active data acquisition, however due to the generality of our approach it also applies to a wide variety of 3D reconstruction problems. With the evaluation done in simulation and on a real robotic platform, exploring and acquiring data from different environments, we demonstrated the generality and usefulness of our approach for next best view estimation and autonomous data acquisition.

Second, in Chapter 4 we presented an active multi-view object recognition approach and describe an information-theoretic framework that combines and unifies two common techniques: online feature selection for reducing computational costs and view planning for resolving ambiguities and occlusions. Our algorithm adaptively chooses between the two strategies of either selecting features that are most informative or moving to a new viewpoint that optimally reduces the expected uncertainty on the identity of the object. This two step process allows us to keep overall computation cost minimal but simultaneously increase recognition accuracy. Extensive evaluation on a large RGB-D dataset with two different feature sets have validated the effectiveness of the proposed framework. Our experiments show that dynamic feature selection alone reduces the computation time at runtime 2.5 to 6 times and, when combining it with viewpoint selection, we significantly increase the recognition accuracy on average by 8–18% absolute compared to systems that do not use these two strategies. By establishing a link between active object recognition and change detection, we were further able to use our framework for the follow-up task of actively detecting object change. Furthermore, we have successfully demonstrated the

framework's applicability to a low-powered quadcopter platform with limited operating time.

Finally, in Chapter 5 we presented an online trajectory optimization approach that optimizes a trajectory such that object recognition performance is improved. Inspired by prior work, we formulate the optimization as a derivative-free stochastic optimization, allowing us to express the cost function in an arbitrary way. The cost function is defined such that information acquisition of target objects is improved, while simultaneously moving towards the goal point. We show the evaluation of our approach on a quadrotor platform in simulation as well as on a real robot. The results show that by using an online optimization approach recognition accuracy is greatly improved, but more importantly the optimized trajectory reduces the uncertainty of the posterior class distribution greatly. Hence, verifying that the optimized trajectory collects more valuable information.

Through extensive experiments, in simulation and on real robotic systems, this thesis shows the usefulness of adaptive action selection. We have shown that adaptive action selection is a useful tool to increase reliability and efficiency for 3D data acquisition and object recognition. However, the use cases for adaptive approaches reach far beyond what we have shown in this work. We believe that adaptive approaches can be a viable solution to make tasks, the robot needs to executes, more reliable and efficient.

References

Ali, H., & Marton, Z.-C. (2014, September). Evaluation of feature selection and model training strategies for object category recognition. In *Proc. of the ieee/rsj int. conference on intelligent robots and systems* (pp. 5036–5042).

Amanatides, J., & Woo, A. (1987). A fast voxel traversal algorithm for ray tracing. In *Eurographics* (Vol. i, pp. 3–10).

Amigoni, F., & Gallo, A. (2005). A multi-objective exploration strategy for mobile robots. In *International conference on robotics and automation* (Vol. 2005, pp. 3850–3855). IEEE.

Atanasov, N., Sankaran, B., Ny, J. L., Pappas, G. J., & Daniilidis, K. (2014). Nonmyopic View Planning for Active Object Classification and Pose Estimation. *IEEE Trans. on Robotics*, *30*(5), 1078–1090.

Bajcsy, R. (1988, August). Active perception. *Proceedings of the IEEE*, *76*(8), 966–1005.

Besl, P., & McKay, N. (1992). A Method for Registration of 3-D Shapes. In *Transactions on pattern analysis and machine intelligence* (Vol. 14, pp. 239–256). IEEE.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc.

Blaer, P. S., & Allen, P. K. (2007). Data acquisition and view planning for 3-D modeling tasks. In *International conference on intelligent robots and systems* (pp. 417–422). IEEE.

Bo, L., Ren, X., & Fod, D. (2010). Kernel Descriptors for Visual Recognition. In *Advances in neural information processing systems (nips)*.

Bo, L., Ren, X., & Fox, D. (2011). Depth kernel descriptors for object recognition. In *Proc. of the ieee/rsj int. conference on intelligent robots and systems* (pp. 821–826). IEEE.

Bo, L., Ren, X., & Fox, D. (2012). Unsupervised feature learning for rgb-d based object recognition. In *Experimental robotics - the 13th int. symp. on exp. robotics* (pp. 387–402).

Borotschnig, H., Paletta, L., Prantl, M., & Pinz, A. (1998). Active Object Recognition in Parametric Eigenspace. In *Proc. of the british machine vision conference* (pp. 63.1–63.10).

Brown, M., & Lowe, D. G. (2005, June). Unsupervised 3D object recognition and reconstruction in unordered datasets. In *Proc. of the int. conference on 3-d digital imaging and modeling* (pp. 56–63).

Charrow, B., Kahn, G., Patil, S., Liu, S., Goldberg, K., Abbeel, P., ... Kumar, V. (2015). Information-theoretic planning with trajectory optimization for dense 3D mapping. *Robotics: Science and Systems (RSS) XI*, 3–12.

Collet, A., & Srinivasa, S. (2010, May). Efficient multi-view object recognition and full pose estimation. In *Proc. of the ieee int. conf. on robotics and automation* (pp. 2050–2055).

Connolly, C. (1985). The Determination of Next Best Views. In *International conference on robotics and automation* (pp. 432–435). IEEE.

Denzler, J., & Brown, C. M. (2002). Information Theoretic Sensor Data Selection for Active Object Recognition and State Estimation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, *24*(2), 145–157.

Dietterich, T. G. (2000). Ensemble methods in machine learning. In *Proc. of the first int. workshop on multiple classifier systems* (pp. 1–15). Springer-Verlag.

Dille, M. (2013). Search and Pursuit with Unmanned Aerial Vehicles in Road Networks. , 228.

Dunn, E., & Frahm, J.-M. (2009). Next best view planning for active model improvement. *Procedings of the British Machine Vision Conference 2009*, 53.1–53.11.

Ekvall, S., Jensfelt, P., & Kragic, D. (2006, Oct). Integrating Active Mobile Robot Object Recognition and SLAM in Natural Environments. In *Proc. of the ieee/rsj int. conference on intelligent robots and systems* (pp. 5792–5797).

Elfes, A. (1989). Using Occupancy Grids for Mobile Robot Perception and Navigation. *Computer*, *22*, 46—-57.

Finman, R., Whelan, T., Kaess, M., & Leonard, J. J. (2013, September). Toward lifelong object segmentation from change detection in dense RGB-D maps. In *Proc. of the european conf. on mobile robots* (pp. 178–185).

Geiger, B. (2009). Unmanned aerial vehicle trajectory planning with direct methods. (August). Retrieved from `http://www.engr.psu.edu/vlrcoe/theses/Geiger{_}Brian.pdf`

Gonzalez-Banos, H., & Latombe, J.-C. (1998). Planning Robot Motions for Range-Image Acquisition and Automatic 3d Model Construction. In *Proc. of the aaai fall symposium series.*

Gonzalez-Banos, H. H., & Latombe, J.-C. (2002). Navigation Strategies for Exploring Indoor Environments. *The International Journal of Robotics Research*, *21*(10-11), 829–848.

Grabner, H., & Bischof, H. (2006, June). On-line Boosting and Vision. In *Proc. of the ieee computer society conference on computer vision and pattern recognition* (Vol. 1, pp. 260–267).

Gupta, M., Muller, J., & Sukhatme, G. (2015, April). Using Manipulation Primitives for Object Sorting in Cluttered Environments. *IEEE Trans. on Automation Science and Engineering*, *12*(2), 608–614.

Guyon, I., & Elisseeff, A. (2003, March). An Introduction to Variable and Feature Selection. *Journal of Machine Learning Research*, *3*, 1157–1182.

Haner, S., & Heyden, A. (2011). Optimal view path planning for visual SLAM. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *6688 LNCS*, 370–380.

Haner, S., & Heyden, A. (2012). Covariance propagation and next best view planning for 3D reconstruction. In *Ssba* (Vol. 7573 LNCS, pp. 545–556). Stockholm.

Hauser, K., & Ng-thow hing, V. (2010). Fast Smoothing of Manipulator Trajectories using Optimal Bounded-Acceleration Shortcuts. , 2493–2498.

Hollinger, G. A., Englot, B., Hover, F. S., Mitra, U., & Sukhatme, G. S. (2013). Active Planning for Underwater Inspection and the Benefit of Adaptivity. *Int. J. Rob. Res.*, *32*(1), 3–18.

Hollinger, G. A., Mitra, U., & Sukhatme, G. S. (2011). Active Classification: Theory and Application to Underwater Inspection. *CoRR*, *abs/1106.5*.

Holz, D., Basilico, N., Amigoni, F., & Behnke, S. (2010). Evaluating the efficiency of frontier-based exploration strategies. In *Robotics (isr), 2010 41st international symposium on and 2010 6th german conference on robotics (robotik)* (Vol. 1, pp. 1–8). Munich, Germany: VDE.

Holz, D., Nieuwenhuisen, M., Droeschel, D., Stückler, J., Berner, A., Li, J., ... Behnke, S. (2014). Active Recognition and Manipulation for Mobile Robot Bin Picking. In *Gearing up and accelerating crossfertilization between academic and industrial robotics research in europe:* (Vol. 94, pp. 133–153).

Hornung, A., Zeng, B., & Kobbelt, L. (2008). Image selection for improved multi-view stereo. In *Conference on computer vision and pattern recognition, cvpr.* IEEE.

Kahn, G., Sujan, P., Patil, S., Bopardikar, S., Ryde, J., Goldberg, K., & Abbeel, P. (2015). Active exploration using trajectory optimization for robotic grasping in the presence of occlusions. *Proceedings - IEEE International Conference on Robotics and Automation*, *2015-June*(June), 4783–4790.

Kalakrishnan, M., Chitta, S., Theodorou, E., Pastor, P., & Schaal, S. (2011). STOMP: Stochastic trajectory optimization for motion planning. In *International conference on robotics and automation.* IEEE.

Kollar, T., & Roy, N. (2008). Trajectory Optimization using Reinforcement Learning for Map Exploration. *International Journal of Robotics Research*, *27*(2), 175–196.

Krainin, M., Curless, B., & Fox, D. (2011). Autonomous generation of complete 3D object models using next best view manipulation planning. In *International conference on robotics and automation* (pp. 5031–5037). IEEE.

Lai, K., Bo, L., Ren, X., & Fox, D. (2011). A large-scale hierarchical multi-view RGB-D object dataset. In *Proc. of the ieee int. conference on robotics and automation* (pp. 1817–1824). IEEE.

Laporte, C., & Arbel, T. (2006). Efficient discriminant viewpoint selection for active Bayesian recognition. *Int. Journal of Computer Vision*, *68*(3), 267–287.

Lowe, D. G. (2004, November). Distinctive Image Features from Scale-Invariant Keypoints. *Int. Journal of Computer Vision*, *60*(2), 91–110.

Luber, M., Spinello, L., & Arras, K. O. (2011, September). People Tracking in RGB-D Data with On-line Boosted Target Models. In *Proc. of the ieee/rsj int. conference on intelligent robots and systems* (pp. 3844–3849).

Makarenko, A., Williams, S., Bourgault, F., & Durrant-Whyte, H. (2002). An experiment in integrated exploration. In *International conference on intelligent robots and systems* (Vol. 1). IEEE/RSJ.

Marton, Z.-C., Seidel, F., Balint-Benczedi, F., & Beetz, M. (2013, May). Ensembles of strong learners for multi-cue classification. *Pattern Recogn. Lett.*, *34*(7), 754–761.

Massios, N. A., & Fisher, R. B. (1998). A Best Next View Selection Algorithm incorporating a Quality Criterion. *Ninth British Machine Vision Conference*, 780–789.

McEnteggart, Q., & Whidborne, J. (2012). A multiobjective trajectory optimisation method for planning environmentally efficient trajectories. *Proceedings of the 2012 UKACC International Conference on Control, CONTROL 2012*(September), 128–135.

Nüchter, A., Surmann, H., & Hertzberg, J. (2003). Planning robot motion for 3d digitalization of indoor environments. In *International conference on advanced robotics (icar)* (pp. 222–227).

Paletta, L., & Pinz, A. (2000). Active object recognition by view integration and reinforcement learning. *Robotics and Autonomous Systems*, *31*(1), 71–86.

Park, C., Pan, J., & Manocha, D. (2012). ITOMP: Incremental trajectory optimization for real-time replanning in dynamic environments. *Proceedings of the International Conference on Automated Planning and Scheduling*, 207–215.

Peng, H., Long, F., & Ding, C. (2005, August). Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, *27*(8), 1226–1238.

Pito, R. (1999). A solution to the next best view problem for automated surface acquisition. In *Transactions on pattern analysis and machine intelligence* (Vol. 21, pp. 1016–1030). IEEE.

Potthast, C., Breitenmoser, A., Sha, F., & Sukhatme, G. S. (2015). Active Multi-View Object Recognition and Online Feature Selection. In *17th international symposium on robotics research (isrr).*

Potthast, C., & Sukhatme, G. S. (2014, January). A Probabilistic Framework for Next Best View Estimation in a Cluttered Environment. *Journal of Visual Communication and Image Representation*, *25*(1), 148–164.

Potthast, C., & Sukhatme, G. S. (2016). Online Trajectory Optimization to Improve Object Recognition. In *2016 international conference on intelligent robots and systems (iros 2016).* [submitted].

Quigley, M., Conley, K., Gerkey, B., FAust, J., Foote, T., Leibs, J., ... Mg, A. (2009). ROS: an open-source Robot Operating System. In *Icra workshop on open source software.*

Rao, A. V. (2014). *Trajectory Optimization: A Survey.* Springer International Publishing.

Ratliff, N., Zucker, M., Bagnell, J. A., & Srinivasa, S. (2009). CHOMP: Gradient optimization techniques for efficient motion planning. In *International conference on robotics and automation.* IEEE.

Richter, C., Bry, A., & Roy, N. (2013). Polynomial Trajectory Planning for Aggressive Quadrotor Flight in Dense Indoor Environments. *Isrr*(Isrr), 1–16. Retrieved from `http://groups.csail.mit.edu/rrg/papers/ISRR13{_}Richter.pdf`

Rusu, R. B., Bradski, G., Thibaux, R., & Hsu, J. (2010, October). Fast 3D Recognition and Pose Using the Viewpoint Feature Histogram. In *Proc. of the ieee/rsj int. conference on intelligent robots and systems.* IEEE/RSJ.

Rusu, R. B., & Cousins, S. (2011). 3D is here: point cloud library. In *International conference on robotics and automation.* IEEE.

Rusu, R. B., Holzbach, A., Beetz, M., & Bradski, G. (2009). Detecting and segmenting objects for mobile manipulation. In *12th international conference on computer vision workshops, iccv workshops 2009* (pp. 47–54). IEEE.

Schimbinschi, F., Schomaker, L., & Wiering, M. (2015, Dec). Ensemble methods for robust 3d face recognition using commodity depth sensors. In *Ieee symposium series on computational intelligence* (pp. 180–187).

Scott, W. R., Roth, G., & Rivest, J.-F. (2003). View planning for automated three-dimensional object reconstruction and inspection. *ACM Computing Surveys*, *35*(1), 64–96.

Shannon, C. (1948). A mathematical theory of communication. *Bell System Technical Journal, The*, *27*(3), 379–423.

Stachniss, C., & Burgard, W. (2003). Exploring unknown environments with mobile robots using coverage maps. In *Proc. of the international conference on artificial intelligence (ijcai)* (pp. 1127–1132).

Stachniss, C., Grisetti, G., & Burgard, W. (2005). Information Gain-based Exploration Using Rao-Blackwellized Particle Filters. In *Robotics: Science and systems i* (pp. 65–72).

Tarabanis, K., Tsai, R. Y., & Kaul, A. (1996). Computing occlusion-free viewpoints. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, *18*(3), 279–292.

Tarabanis, K. a., Allen, P. K., & Tsai, R. Y. (1995). Survey of sensor planning in computer vision. *Transactions on Robotics and Automation*, *11*(1), 86–104.

Theodorakopoulos, P. (2009). On autonomous target tracking for UAVs.

Thrun, S. (2002). Robotic mapping: A survey. *Exploring artificial intelligence in the new millennium*(February).

Thrun, S., Burgard, W., & Fox, D. (2005). *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents).* The MIT Press.

Tilton, J. C., & Swain, P. H. (1984). Contextual classification of multispectral image data. *Image and Vision Computing*, *2*(1), 13–29.

Trummer, M., Munkelt, C., & Denzler, J. (2010). Online next-best-view planning for accuracy optimization using an extended E-criterion. In *International conference on pattern recognition* (pp. 1642–1645).

Verleysen, M., & Rossi, F. (2009). Advances in Feature Selection with Mutual Information. *CoRR*.

Welke, K., Issac, J., Schiebener, D., Asfour, T., & Dillmann, R. (2010, May). Autonomous acquisition of visual multi-view object representations for object recognition on a humanoid robot. In *Proc. of the ieee int. conference on robotics and automation* (pp. 2012–2019).

Wenhardt, S., Deutsch, B., Hornegger, J., Niemann, H., & Denzler, J. (2006). An information theoretic approach for next best view planning in 3-D reconstruction. In *International conference on pattern recognition* (Vol. 1, pp. 103–106).

Wettach, J., & Berns, K. (2010). Dynamic Frontier Based Exploration with a Mobile Indoor Robot. In *Robotics (isr), 2010 41st international symposium on and 2010 6th german conference on robotics (robotik)* (pp. 1–8).

Yamauchi, B. (1997). A frontier-based approach for autonomous exploration. In *International symposium on computational intelligence in robotics and automation cira'97. 'towards new computational principles for robotics and automation'.* IEEE.

Zhou, X. S., Comaniciu, D., & Krishnan, A. (2003, Oct). Conditional feature sensitivity: a unifying view on active recognition and feature selection. In *Proc. of the ieee int. conference on computer vision* (p. 1502-1509 vol.2).