TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

TUOMAS VÄLIMÄKI

OPTIMIZING GAZE DIRECTION IN A VISUAL NAVIGATION TASK

Master of Science Thesis

Examiner: Professor Risto Ritala
Examiner and topic approved by the
Faculty Council of the Faculty of
Engineering Sciences on
9th September 2015

ABSTRACT

TAMPERE UNIVERSITY OF TECHNOLOGY
Master's Degree Programme in Automation Engineering
TUOMAS VÄLIMÄKI: Optimizing gaze direction in a visual navigation task
Master of Science Thesis, 42 pages
November 2015
Major: Systems Analysis
Examiner: Professor Risto Ritala
Keywords: planning under uncertainty ,POMDP, active sensing

Navigation in an unknown environment consists of multiple separable subtasks, such as collecting information about the surroundings and navigating to the current goal. In the case of pure visual navigation, all these subtasks need to utilize the same vision system, and therefore a way to optimally control the direction of focus is needed. This thesis presents a case study, where the active sensing problem of directing the gaze of a mobile robot with three machine vision cameras is modeled as a partially observable Markov decision process (POMDP) using a mutual information (MI) based reward function. The key aspect of the solution is that the cameras are dynamically used either in monocular or stereo configuration.

The algorithms are implemented on Robot Operating System (ROS) and the benefits of using the proposed active sensing implementation over fixed stereo cameras are demonstrated with simulations experiments. The proposed active sensing outperforms the fixed camera solution when prior information about the environment is highly uncertain, and performs just as good in other tested scenarios.

TIIVISTELMÄ

TAMPEREEN TEKNILLINEN YLIOPISTO
Automaatiotekniikan koulutusohjelma
TUOMAS VÄLIMÄKI: Kamerajärjestelmän suunnan optimointi navigointitehtävässä
Diplomityö, 42 sivua
Marraskuu 2015
Pääaine: Systeemien analysointi
Tarkastaja: professori Risto Ritala
Avainsanat: päätöksenteko epävarmuuden vallitessa ,POMDP, kameroiden aktiivinen operointi

Navigaatio ennalta tuntemattomassa ympäristössä koostuu useista erillisistä alitehtävistä kuten informaation keräämisestä ja tämänhetkiseen kohteeseen navigoinnista. Kun kyse on puhtaasti visuaalisesta navigoinnista, tarvitsee kaikkien alitehtävien hyödyntää samaa kamerajärjestelmää, joten kamerajärjestelmän suunnan optimointi on tarpeen. Tässä diplomityössä esitellään esimerkkitapaus, jossa kolmen mobiiliin robottiin kiinnitetyn kameran suunnan aktiivinen operointiongelma mallinnetaan osittain havaittavana Markov-päätösprosessina (POMDP), jossa käytetään keskinäisinformaatioon (MI) perustuvaa palkkiota. Olennainen osa ratkaisua on, että kameroita voidaan käyttää dynaamisesti sekä monokulaarisessa- että stereokamera-konfiguraatiossa.

Kehitetyt algoritmit implementoidaan Robot Operating System (ROS) -järjestelmälle ja kameroiden aktiivisen operoinnin hyödyt verrattuna kiinteästi asennettuihin stereokameroihin osoitetaan simulaatioilla. Kehitetty aktiivinen operointi suoriutuu kiinteitä kameroita paremmin kun ennakkotieto ympäristöstä on hyvin epävarmaa, ja muissa kokeilluissa tapauksissa vähintään yhtä hyvin.

PREFACE

This work was carried out in the Measurement Information research group at Department of Automation Science and Engineering, Tampere University of Technology, during April 2015 – November 2015. The grant by Industrial Research Fund of Tampere University of Technology is gratefully acknowledged.

I would like to thank Prof. Ritala for providing me with the opportunity to work in his research group, and all my colleagues for their input and support. My special thanks go to Mikko Lauri for his helpful comments and suggestions.

I would also like to thank all my friends and family for supporting me and for bringing so much joy into my life.

Tampere, 14$^{\text{th}}$ November, 2015

Tuomas Välimäki

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS AND ABBREVIATIONS

EKF	Extended Kalman filter
GMM	Gaussian mixture model
IMU	Inertial measurement unit
KL	Kullback-Leibler, (divergence)
MDP	Markov decision process
MI	Mutual information
PBVI	Point-based value iteration
pdf	Probability density function
PID	Proportional-integral-derivative, (controller)
POMDP	Partially observable Markov decision process
PWLC	Piecewise linear and convex
QR	Quick response, (code), a two-dimensional bar-code
ROS	Robot Operating System, an open source framework for robotics software development
SLAM	Simultaneous localization and mapping

$\cup$	Union of arbitrary sets
$\oplus$	Cross-sum between sets
α	Vector containing the value of a policy tree for all states
β	Camera angle of view
$\Gamma(b, a, b')$	Belief transition function
γ	Discount factor
δ	Measured angle
ϵ	Magnitude of the Bellman error
θ_t	Robot's heading
ϑ	Weighing factor for information gain on current target
κ	Constant positive scalar penalty
Λ	Set of all α-vectors
μ	Mean of belief
π	Policy
$\rho(b, a)$	Reward for executing action a in belief b
Σ	Covariance of belief
$\tau(b, a, z')$	Belief update function
φ	Weighing factor for information gain on next target
χ_n	The best n-step α-vector
ω_t	Rotational velocity
A	Action space

a	Action
B	Belief space
$b(s)$	Belief
$C(a_t, \beta, r_{max})$	Camera cone of observation
d_n	Decision rule when there are n steps remaining
$D_{KL}(\cdot, \cdot)$	Kullback-Leibler divergence
$\mathbb{E}[\cdot]$	Expected value
$f(\cdot)$	Transition function
$g(a_t, a_{t-1})$	Cost for taking action a_t when previous action was a_{t-1}
H	Planning horizon
h_t	History of actions and measurements up to time t
$\mathcal{I}(\cdot, \cdot)$	Mutual information
K_d	PID controller's derivative gain
K_i	PID controller's integral gain
K_p	PID controller's proportional gain
L^i	Belief about location of i^{th} landmark
l^i	Location of i^{th} landmark
$\ln(\cdot)$	Natural logarithm
$m(\cdot)$	Measurement function
$\mathcal{N}(\mu, \Sigma)$	Multivariate normal distribution with mean μ and covariance Σ
$O(s', a, z')$	Observation function
$p(X)$	Probability of X
Q	Covariance of control noise
q_t	Control noise
$R(s, a)$	Reward for executing action a in state s
$\mathbb{R}$	Set of real numbers
r_{max}	Camera maximum range
S	State space
s	State
s_t^r	Robot's state at time t
$T(s, a, s')$	State transition model
t	Time
$\text{tr}(\cdot)$	Trace
u_t	Robot control
V^π	Value of a stationary policy π
$VA(C_{j=1:3})$	Area from which measurements are obtained
V_n^π	Value of policy π when there are n steps remaining
v_t	Translational velocity
W	Covariance of measurement noise

w_t	Measurement noise
x_t	Robot's x-coordinate
y_t	Robot's y-coordinate
Z	Observation space
z	Observation

1. INTRODUCTION

In human vision, the sequential deployment of gaze in multiple directions is a vital part of information gathering (Johnson et al. 2014). If a person is given a task to follow a set of driving instructions, they seemingly effortlessly navigate to the desired destination, while simultaneously observing their surrondings. The instructions could be a sequence of landmarks, e.g. drive straight until you see a yellow house, then turn left and continue until you reach a house with a white picket fence. Similar approaches can be used in mobile robotics, however, there is little research done on actively manipulating individual sensors in a vision system to optimize information gathering.

Active sensing in general refers to seeking a policy for determining the optimal sensor configuration at each time instance as a function of information from previous measurements to achieve a goal (Hero and Cochran 2011). The problem can crudely be categorized as an information-gathering problem or as a task-achievement problem depending on whether the goal is to gather maximum amount of information or merely the completion of a task not related to sensing itself. This thesis focuses on the former.

Robots face uncertainty both in predicting and sensing their own state and that of the surroundings. Regardless of whether the goal is to gather information or achieve a task, the active sensing problem becomes a sequential decision making problem in a stochastic environment and can therefore be modeled as a partially observable Markov decision process (POMDP) (Kaelbling, Littman, and Cassandra 1998). The utility of actions is measured by a reward function and the robot acts so as to maximize the expected reward. Because the state is not directly observable, the robot's knowledge of it is described as a probability density function (pdf) over the state known as the belief state.

Typical information-gathering problems in robotics are exploration tasks. Controlling the direction of sensor focus independently of the robot pose is beneficial when navigating in unknown environments, especially on car-like robots that cannot turn on spot, to gather information about the surroundings. Nevertheless, often only the pose of the robot is considered, and the sensor placements are fixed. Stachniss, Grisetti, and Burgard (2005) propose a solution for integrated simultaneous localization

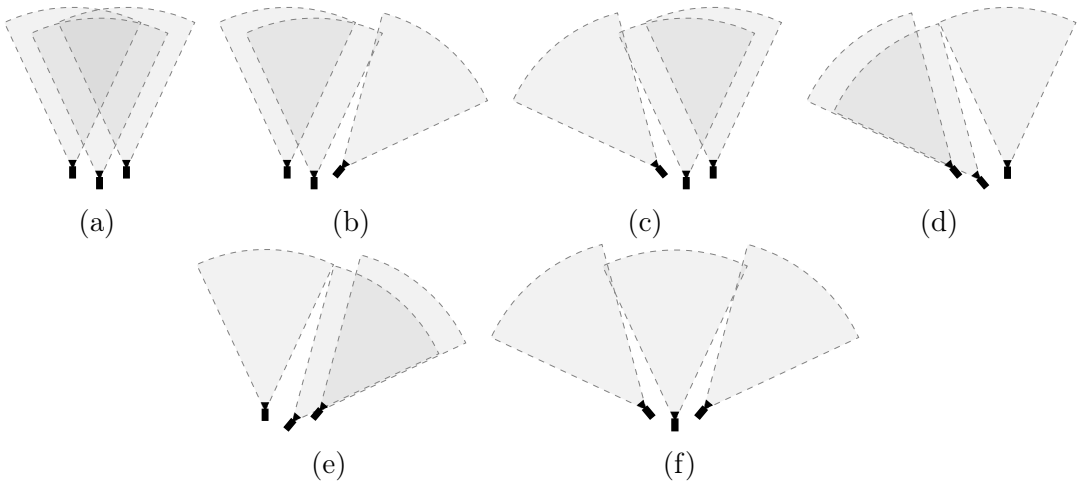


Figure 1.1 *The 6 camera modes of the proposed system.*

and mapping (SLAM) and exploration using a Rao-Blackwellized particle filter to compute the expected information gain of actions. The approach can be regarded as an approximation of a myopic POMDP.

Often the active sensing problems are modeled as task-achievement problems, where the reward function depends only on state and actions and is therefore linear in the belief state and can be solved using standard POMDP solvers. Such solutions have been presented e.g. by Spaan and Lima (2009) and Spaan, Veiga, and Lima (2014). In pure information-gathering problems it is natural to use information theoretic rewards that are nonlinear in the belief state. For example Araya-López et al. (2010) introduce several rewards typically used for maximizing information, such as the Kullback-Leibler (KL) divergence. Other nonstandard POMDPs have also been proposed e.g. by Krishnamurthy and Djonin (2007) and Martinez-Cantin et al. (2009).

Lauri and Ritala (2014) recently discuss an active sensing problem, where a robot must decide the direction of focus of its vision system while moving along a fixed trajectory, and propose a solution via an approximate open-loop feedback control. Raunio, Välimäki, and Ritala (2015) present a case, where a robot is set to follow a target in an environment that contains additional moving objects. The direction of its camerasystem is operated to maximize information about distance to target, and about the location of the additional moving object to avoid collision.

This thesis studies a similar case as Lauri and Ritala, but instead of the trajectory being fixed, it consists of visual landmarks at uncertain locations. The robot must select the direction of gaze of its vision system while navigating the trajectory in order to maximize the amount of information about the positions of the current and next landmark in sequence. The problem is formulated as a myopic POMDP with information-theoretic reward.

The vision system consists of three cameras. The cameras can be in 6 discreet modes as seen in Figure 1.1 to either maximize the field of view or to focus two or even three cameras in the same direction. When any two cameras face the same direction, they are used as a stereo pair to allow depth computation as opposed to a monocular camera, where it is only possible to measure the relative angle of the landmark seen in the image with respect to the camera. However, using the cameras in a monocular configuration is beneficial when searching for a landmark to narrow down its possible location, which can then be refined by using a stereo pair.

The main contribution of this thesis is to demonstrate the benefits of using the cameras dynamically in either monocular or stereo configurations depending on the expected information gain, as opposed to a fixed stereo configuration, where the cameras are permanently in mode (a) of Figure 1.1. The fixed camera configuration and the developed active sensing implementation are compared using simulation experiments. Furthermore, the developed implementation is intended to be applied in a real-world scenario with an actual four-wheeled mobile robot.

The remainder of the thesis is organized as follows. In Chapter 2 the basic principles of Markov decision processes, partially observable Markov decision processes, and both Kalman filters and extended Kalman filters are outlined. Chapter 3 formulates the case problem. The solution and implementation is described in Chapter 4 and the results of the simulation experiments are reported in Chapter 5. Chapter 6 introduces future experiments to be carried out with an actual robot and Chapter 7 concludes the thesis.

2. PRELIMINARIES

2.1 Markov decision processes

Let us consider a sequential decision making process, where $s \in S$ denotes the state of the system at any time t. At each time step the agent takes an action $a \in A$ and receives a reward $R : S \times A \to \mathbb{R}$, where $R(s, a)$ is the expected reward of executing action a in state s. $T : S \times A \times S \to [0, 1]$ is the state transition model, where $T(s, a, s') = p(s'|s, a)$ describes the probability of ending in state s' when taking the action a in state s. In this model, the next state and the expected reward depend only on the current state and the action taken, which is called the Markov property. This type of decision processes are called Markov decision processes (MDPs) and are presented as the tuple $\langle S, A, T, R\rangle$ (Bellman 1957; Puterman 1994; White 1993). For now, let us assume a finite set of states and actions.

Although $R(s, a)$ provides a measure of the immediate reward at each time step, usually we are more interested in the amount of reward gained during the whole decision making problem. Expected future discounted reward,

$$\mathbb{E}\left[\sum_{t=0}^{H-1} \gamma^t R(s_t, a_t)\right], \tag{2.1}$$

provides a trade-off between immediate and future reward, and is often used in MDP literature. H is the possibly infinite horizon of the problem, and $\gamma \in [0, 1)$ is a discount factor to give less value to rewards received in the future. The aim of MDP solving is to maximize this quantity.

A policy is a sequence of decision rules, $\pi = [d_0, d_1, \ldots, d_{H-1}]$, an action-selection strategy where each decision rule is a mapping from states to actions: $d_t : S \to A$. Finding the optimal policy for a given horizon is called solving the MDP. Since the state and actions spaces are finite, a decision rule can be represented with a finite length vector of size $|S|$, and there is a finite number of decision rules $|A|^{|S|}$. This results in a large but finite number of finite horizon policies (Cassandra 1998).

If there is a different decision rule for each time step, the policy is called non-stationary, which is often required for finite horizon problems to act optimally. The

agent should probably act differently if it only has one decision step remaining than when it has multiple. For this reason, it is convenient to define the problem as how many time steps are remaining until the end of the horizon rather than what the current time step is. In an infinite horizon problem the policy becomes stationary: at each time step the agent has an infinite amount of decision steps remaining. As shown by Howard (1960), for an infinite horizon problem there exists a stationary policy $\pi = [d, d, d, \dots]$ that is optimal for all starting states.

The utility of actions, and policies, is determined by value functions, $V : S \to \mathbb{R}$. Using Bellman's principle of optimality (Bellman 1957) and dynamic programming, the value of starting in state s and following a policy π when there are n steps remaining can be calculated recursively as

$$V_n^\pi(s) = R(s, d_n(s)) + \gamma \sum_{s' \in S} T(s, d_n(s), s') V_{n-1}^\pi(s'), \tag{2.2}$$

where the dynamic programming indice i.e. steps remaning, n, goes backwards in time: $t = 0 \Leftrightarrow n = H$, $t = H - 1 \Leftrightarrow n = 1$, and $V_0^\pi(s) = 0$ for all states s.

The value of a stationary policy over an infinite horizon is

$$V^\pi(s) = R(s, d(s)) + \gamma \sum_{s' \in S} T(s, d(s), s') V^\pi(s'), \tag{2.3}$$

and has a unique solution (Bellman 1957). Evaluating a stationary policy over a finite horizon is an approximation of Equation 2.3 as

$$\lim_{n \to \infty} \| V_n^\pi(s) - V^\pi(s) \| = 0, \tag{2.4}$$

which is shown e.g. by Bertsekas (1995).

Dynamic programming allows for the computation of an optimal policy, where

$$V_n^*(s) = \max_{a \in A} \left[R(s, a) + \gamma \sum_{s' \in S} T(s, a, s') V_{n-1}^*(s') \right] \tag{2.5}$$

is the value of optimal policy π^* when starting in state s and there are n steps remaining. The recursive computation of the optimal value function is called value iteration, where a single computation step is referred to as the Bellman update. As a convenience notation, we can write this with the help of Q-functions:

$$V_n^*(s) = \max_{a \in A} Q_n(s, a), \tag{2.6}$$

where

$$Q_n(s,a) = R(s,a) + \gamma \sum_{s' \in S} T(s,a,s') V^*_{n-1}(s'), \tag{2.7}$$

i.e. $Q_n(s,a)$ is the value of taking action a when there are n steps remaining and acting optimally for the remaining $n-1$ steps.

Now we can simultaneously compute the optimal policy $\pi^* = [d^*_H, d^*_{H-1}, \ldots, d^*_0]$ using

$$d^*_n(s) = \arg\max_{a \in A} Q_n(s,a). \tag{2.8}$$

The optimal infinite horizon value,

$$V^*(s) = \max_{a \in A} \left[R(s,a) + \gamma \sum_{s' \in S} T(s,a,s') V^*(s') \right], \tag{2.9}$$

is proven to converge within some ϵ of the optimal values in a finite number of iterations (Puterman 1994).

MDPs can be solved using the value iteration introduced above, and Papadimitriou and Tsitsiklis (1987) have analysed the complexity of solving such problems. However, MDPs assume that the state is completely observable and known to the agent, which is often not the case in real world applications.

2.2 Partially observable Markov decision processes

2.2.1 POMDP formulation

In case the observations about the state are imperfect and noisy, we model the problem as a partially observable Markov decision process (POMDP) (Ross et al. 2008; Smallwood and Sondik 1973; Sondik 1978). POMDPs are formally presented as the tuple $\langle S, A, T, R, Z, O \rangle$, where S, A, T, and R are those already defined for MDP. Let us define Z as the set of all possible observations and $O : S \times A \times Z \to [0,1]$ as the observation function, where $O(s', a, z') = p(z'|a, s')$ is the probability of observing z' if action a is performed and the resulting state is s'.

Because the states are not directly observable, the agent cannot choose its actions based on the state only. Instead it has to consider the whole history of actions and observations $h_t = \{a_0, z_1, \ldots, a_{t-1}, z_t\}$. Maintaining all this information would be memory expensive, but the belief state

$$b(s) = p(s_t = s | h_t, b_0), \tag{2.10}$$

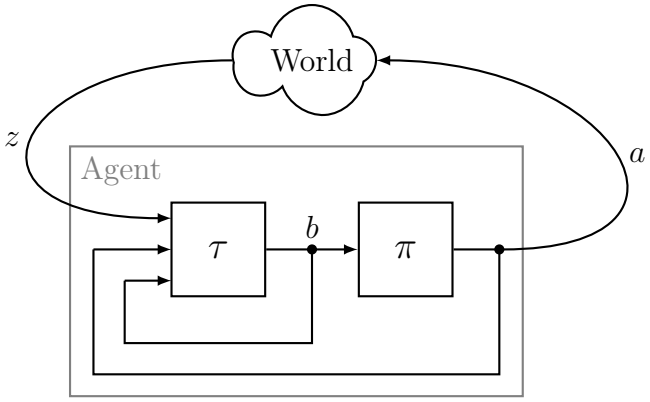


Figure 2.1 *POMDP problem structure. A POMDP agent can be decomposed to a state estimator (τ) and a policy (π). Figure adapted from Kaelbling, Littman, and Cassandra (1998).*

where b_0 is the initial belief about the starting state, is a sufficient statistic for the history (Smallwood and Sondik 1973).

The next belief state $b(s')$ can be calculated from the previous belief state $b(s)$, previous action a, and current observation z' using the belief update function $\tau(b, a, z')$ given by the Bayes' rule

$$\begin{aligned} b(s') &= \tau(b, a, z')(s') \\ &= \frac{1}{p(z'|b, a)} O(s', a, z') \sum_{s \in S} T(s, a, s') b(s), \end{aligned} \tag{2.11}$$

where

$$p(z'|b, a) = \sum_{s' \in S} O(s', a, z') \sum_{s \in S} T(s, a, s') b(s) \tag{2.12}$$

is the prior probability of observing z' and acts as the normalizing term. Essentially, POMDPs can be decoupled into a state estimator of the form presented in Equation 2.11 and a policy π defining the actions, as illustrated in Figure 2.1.

As with MDPs the objective of POMDP planning is to find policies according to which the agent should take actions. Because in a partially observable case, we do not have access to the system states, the policy takes a more complicated form than just a direct mapping from states to actions. Let us first consider finite horizon policies. When an agent has one step remaining, its only option is to select an action. With two steps remaining, it can select an action, make an observation, and based on the observation take another action. For a finite horizon H the policy is a tree of height H as seen in Figure 2.2. The top node of the policy tree designates the first action, and the observation received determines the branch to follow i.e. the next action to take. The full n-step policy tree describes the complete policy.

The value of executing a policy tree depends on the starting state s. As a tree of

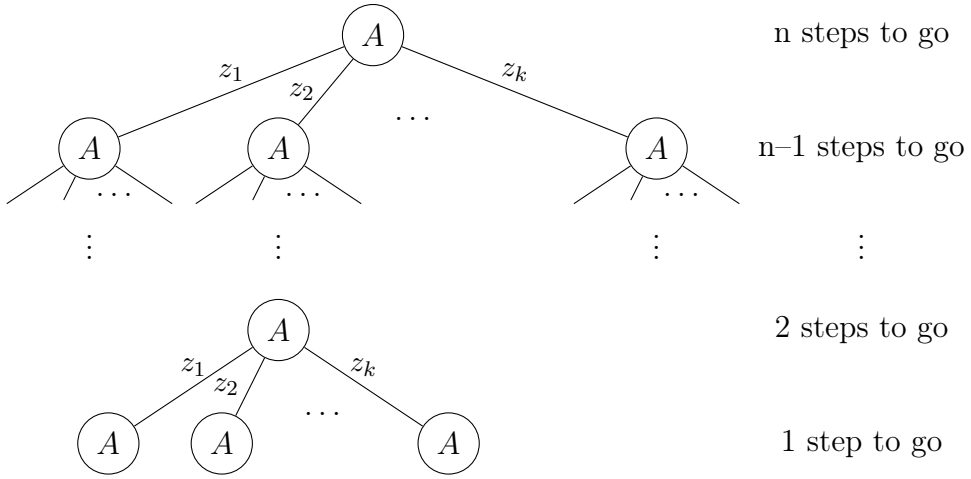


Figure 2.2 *An n-step policy tree describes the complete policy for n steps of conditional behaviour. The top node determines the first action $a \in A$, and depending on the observation $z \in Z$ the next action node is reached.*

height n can be represented recursively by its subtrees of height $n-1$, also the value of a tree can be expressed recursively. Because the exact state is not known to the agent, the value of executing a policy tree must be determined from the belief state as the expectation over the actual states

$$V_n^\pi(b) = \sum_{s \in S} b(s) V_n^\pi(s), \tag{2.13}$$

and $V_{n-1}^\pi(s')$ is conditioned by the probability of the observation connecting the nodes, i.e.

$$\begin{aligned} V_n^\pi(s) &= R(s, a_\pi) + \gamma \sum_{s' \in S} T(s, a_\pi, s') \sum_{z' \in Z} O(s', a_\pi, z') b(s') V_{n-1}^\pi(s') \\ &= R(s, a_\pi) + \gamma \sum_{s' \in S} T(s, a_\pi, s') \sum_{z' \in Z} O(s', a_\pi, z') \tau(b, a_\pi, z') V_{n-1}^\pi(s'). \end{aligned} \tag{2.14}$$

The action stated in the top node of the tree is denoted with a_π.

A commonly used method in deriving the optimal value function is the use of α-vectors. Let α^π be a vector of size $|S|$ that consists of values of the policy tree π for each state s, when the action stated by the top node of the policy tree is a_π: $\alpha^\pi = [V_n^\pi(s_0), V_n^\pi(s_1), \ldots, V_n^\pi(s_N)]$. To determine the optimal n-step value function, the best policy tree for a given belief state is selected:

$$\begin{aligned} V_n^*(b) &= \max_{\alpha \in \Lambda_n} \sum_{s \in S} b(s) \alpha(s) \\ &= \max_{\alpha \in \Lambda_n} b \cdot \alpha, \end{aligned} \tag{2.15}$$

where Λ_n is the set of all n-step α-vectors corresponding to the finite set of n-step

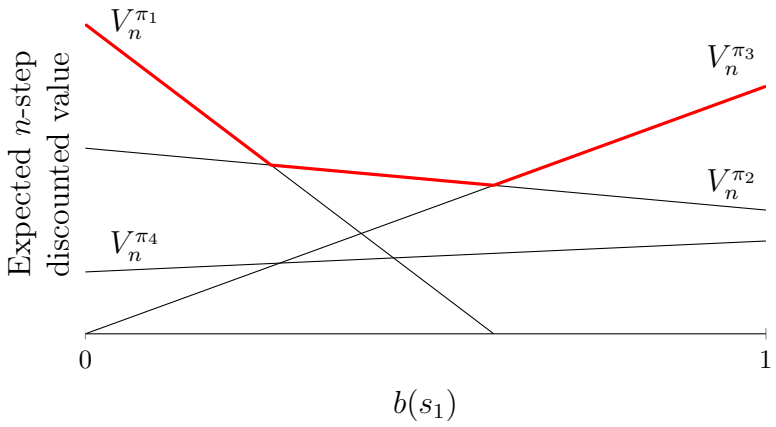


Figure 2.3 *The optimal n-step value function is the upper surface of the value functions of all n-step policy trees. The PWLC property of the optimal value function is apparent.*

policy trees. The optimal action to take when there are n steps remaining is the action associated with the best α-vector.

Equations 2.13 and 2.15 lead to an important property of the optimal n-step value function. Each policy tree produces a value function V_n^π linear in b, and V_n^* is the upper surface of this collection of functions. Therefore V_n^* is piecewise linear and convex (PWLC) (Smallwood and Sondik 1973). This property can be clearly illustrated for a state space of size two, as done in Figure 2.3. As the belief space forms a simplex, $b(s_2) = 1 - b(s_1)$, a single number is enough to describe the whole space. The value functions of policy trees π_i, $V_n^{\pi_i}$, are linear in b and represented as lines in the figure. The optimal value function V_n^* is the maximum of all $V_n^{\pi_i}$ at each point in the belief space, and therefore forms the upper surface of the functions.

The bound on accuracy when approximating the optimal infinite horizon value function V^* with n iterations can be expressed in terms of the magnitude of the Bellman error $\epsilon = \|V_n - V_{n-1}\|$. When V_n^* is the optimal n-step value function, then $\|V_n^* - V^*\| \leq \frac{2\epsilon}{1-\gamma}$ (Hauskrecht 2000).

2.2.2 Exact value iteration

Combining the Equations 2.13, 2.14, and 2.15, the Bellman update for POMDPs can be done using

$$V_n^*(b) = \max_{a \in A} \sum_{s \in S} \sum_{z' \in Z} b(s) \left[\frac{R(s,a)}{|Z|} + \gamma \sum_{s' \in S} T(s,a,s') O(s',a,z') \chi_{n-1}(\tau(b,a,z'),s') \right],$$

where $\chi_n(b,s)$ is the s^{th} element of the vector $\chi_n(b) = \arg\max_{\alpha\in\Lambda_n} b\cdot\alpha$. The term in brackets generates $|Z|\times|A|$ Λ-sets, each one of size $|\Lambda_{n-1}|$, defined as

$$\overline{\Lambda_n}^{a,z'} = \left\{ \frac{R^a}{|Z|} + P^{a,z'}\cdot\alpha_{n-1} \middle| \alpha_{n-1}\in\Lambda_{n-1} \right\},$$

where $P^{a,z'}(s,s') = T(s,a,s')O(s',a,z')$ and $R^a(s) = R(s,a)$. For the exact representation of the value function, we would need to compute

$$\overline{\Lambda_n} = \bigcup_{a\in A}\bigoplus_{z'\in Z} \overline{\Lambda_n}^{a,z'},$$

where $\oplus$ denotes the cross-sum between two sets, and $\cup$ the union of arbitrary sets (Araya-López et al. 2010).

A policy tree for a horizon of length n contains $\sum_{t=0}^{n-1}|Z|^t = \frac{|Z|^n-1}{|Z|-1}$ nodes, where in each node there are $|A|$ choices of actions. Therefore, the size of the set of all possible n-step policy trees, i.e. the size $|\Lambda_n|$, is $|A|^{\frac{|Z|^n-1}{|Z|-1}}$.

As seen from Figure 2.3, all policy trees do not necessarily contribute to the optimal value function; In this case $V_n^{\pi_4}$ is dominated by other value functions in all parts of the belief space. Often the set of all possible n-step α-vectors, and the assosiated policy trees, can be pruned to form a parsimonious subset Λ^- that represents the same optimal value function:

$$\begin{aligned} V_n^*(b) &= \max_{\alpha\in\Lambda_n} b\cdot\alpha \\ &= \max_{\alpha\in\Lambda_n^-} b\cdot\alpha, \end{aligned} \tag{2.16}$$

thus limiting the size of Λ-sets that need to be calculated for the exact value iteration.

In a parsimonious set all α-vectors and corresponding policy trees are useful (Kaelbling, Littman, and Cassandra 1998). Different value based POMDP methods differ in the way they form the parsimonious set Λ_n^-. Still, exact value iteration is highly computationally complex and infeasible for anything but small scale problems, and further, Littman (1996) showed that identifying all useful α-vectors is an intractable problem by its self.

2.2.3 Approximate solutions

Due to the computational complexity of an exact solution, many approximate POMDP solutions have been developed. Hauskrecht (2000) provides a comprehensive survey of different value function approximations. The intractability of the exact

solutions is mainly a consequence of computing an optimal policy over the entire belief space (Roy, Gordon, and Thrun 2005), and many approximation methods aim to somehow compress the belief space to a discrete subset of belief points. These methods are called point-based approximations and rely on point-based value iteration (PBVI).

Point-based methods have largely contributed to the recent progress in solving POMDPs and the original PBVI algorithm (Pineau, Gordon, and Thrun 2003) has later been refined (Pineau, Gordon, and Thrun 2006), and other point-based algorithms such as Perseus (Spaan and Vlassis 2005) and SARSOP (Kurniawati, Hsu, and Lee 2008) have been developed.

A typical point-based algorithm, at each iteration n until convergence:

1. selects a new set of belief points B_n based on B_{n-1} and the current approximation V_{n-1},
2. performs a Bellman update at each belief point $b \in B_n$ resulting in one α-vector per point,
3. prunes points whose α-vectors are dominated or considered negligible.

The different algorithms differ mainly in how new belief points are selected and how the update is performed. Shani, Pineau, and Kaplow (2013) have done a large survey of point-based POMDP solvers and their differences.

Suboptimal myopic policies are also widely used due to the intractability of an optimal solution (Ji, Parr, and Carin 2007; Kreucher, Hero, and Kastella 2005; Stachniss, Grisetti, and Burgard 2005). In a myopic policy, only the immediate reward for taking an action is considered i.e. the optimization horizon $H = 1$. Using myopic policies can, however, be a very poor approximation. In active sensing objects can reside in sensor range only for a brief moment if e.g. the environment is non-stationary or the objects get occluded behind obstacles. Therefore acting greedily and selecting actions that maximize only the immediate reward can perform poorly over time.

2.2.4 Extension to belief dependent rewards

POMDPs can be reformulated as a MDP over the belief space, $\langle B, A, \Gamma, \rho \rangle$, where $B = \Pi(S)$ is the belief space (the set of probability distributions over S), and Γ is the belief transition model defined as

$$\Gamma(b, a, b') = p(b'|b, a) = \sum_{z' \in Z} p(b'|b, a, z')p(z'|b, a), \tag{2.17}$$

where

$$p(b'|b,a,z') = \begin{cases} 1 & \text{if } \tau(b,a,z') = b' \\ 0 & \text{otherwise.} \end{cases} \tag{2.18}$$

With this reformulation, a number of results for MDPs can be extended to POMDPs, such as the existence of a deterministic optimal policy (Araya-López et al. 2010). However, even if the POMDP has a finite state space, the belief MDP is defined over a continuous i.e. infinite belief space, which complicates the solving compared to a discrete state MDP.

The objective of this belief-MDP is to find a policy that maps belief states $b \in B$ to actions; $\pi : B \to A$, so as to maximize the expected discounted future reward, $\mathbb{E}\left[\sum_{t=0}^{H-1} \gamma^t \rho(b_t, a_t)\right]$, where γ is the discount factor, H the possibly infinite time horizon, and $\rho(b_t, a_t)$ the reward function defining the immediate reward for executing action a_t in belief state b_t.

The Bellman update over a continuous space can be written

$$\begin{aligned} V_n^*(b) &= \max_{a \in A} \left[\rho(b,a) + \gamma \int_{b' \in B} \Gamma(b,a,b') V_{n-1}^*(b') \mathrm{d}b' \right] \\ &= \max_{a \in A} \left[\rho(b,a) + \gamma \sum_{z' \in Z} p(z'|b,a) V_{n-1}^*(\tau(b,a,z')) \right], \end{aligned} \tag{2.19}$$

where for all $b \in B, V_0(b) = 0$.

In a standard POMDP formulation the reward function $\rho(b,a)$ would be defined as the expected reward of the underlying MDP, $\sum_{s \in S} b(s) R(s,a)$. With this definition, and using Equation 2.12, Equation 2.19 could be written as

$$V_n^*(b) = \max_{a \in A} \sum_{s \in S} b(s) \left[R(s,a) + \gamma \sum_{s' \in S} T(s,a,s') \sum_{z' \in Z} O(s',a,z') \tau(b,a,z') V_{n-1}^*(s') \right],$$

which is consistent with the value function presented in Equations 2.13 and 2.14.

However, when the objective of POMDP planning is to maximize the quality of information, which is often the case in active sensing, sensor selection, and surveillance problems, information theoretic rewards that depend directly on the belief, and not on the underlying state provide a natural measure of the utility of actions. Araya-López et al. (2010) present several such rewards, i.e. the Kullback-Leibler (KL) divergence, and Kreucher, Hero, and Kastella (2005) compare the performance differences of using either information theoretic or state dependend rewards in sensor management.

The problem with information theoretic rewards is, that the value function is no longer guaranteed to remain PWLC. As most available standard POMDP solvers

exploit the PWLC property of the value function, this makes the solving more difficult. However, Araya-López et al. (2010) show that if $\rho(b, a)$ is convex, the value function is convex. Further, if $\rho(b, a)$ is PWLC and $V_0(b) = 0$, then V_n^* is PWLC. If $\rho(b, a)$ is not PWLC they propose approximating it with a PWLC function, and show that the algorithms used to solve the problem converge to the optimal value function.

Other solutions for belief dependent rewards are e.g. Spaan, Veiga, and Lima's (2014) IR-POMDP that averts the problem by introducing specific information actions to the decision space and keeping the reward function linear in the belief state. Krishnamurthy and Djonin (2007) apply structured threshold policies and use stochastic approximation algorithms to calculate the best policy, whereas Chong, Kreucher, and Hero (2008) present a Monte-Carlo approximation method for POMDPs that does not rely on the PWLC property.

The problem introduced by the use of information theoretic rewards can also be averted by using myopic policies—with the cost of being suboptimal. In myopic solutions no value iteration is performed, and therefore the PWLC property of the value function is irrelevant. Myopic policies are widely used in sensor management, even though they have limitations as explained at the end of Section 2.2.3.

2.2.5 Extension to continuous state and observation spaces

Although navigation is fundamentally a continuous problem, the majority of POMDP research, as well as most of the algorithms and solutions introduced in the previous sections, have focused on discrete problems. For a continuous state space the belief update function presented in Equation 2.11 is of form

$$\tau(b, a, z')(s') = \frac{1}{p(z'|b, a)} O(s', a, z') \int_{s \in S} T(s, a, s') b(s) \mathrm{d}s, \tag{2.20}$$

where

$$p(z'|b, a) = \int_{s' \in S} O(s', a, z') \int_{s \in S} T(s, a, s') b(s) \mathrm{d}s \, \mathrm{d}s'. \tag{2.21}$$

For a continuous state POMDP, the corresponding belief space is the infinite-dimensional functional space defined by the set of pdfs over the state space. This added complexity is one of the reasons why POMDP research has focused on the discrete-state case (Porta et al. 2006).

To apply discrete POMDP algorithms to problems with continuous states and observations, a common approach is to discretize the state and observation spaces. This can lead to poor performance if the discretization is not fine enough, whereas

fine discretization causes computability issues. Since the dimensionality of the belief space equals the number of states, computational cost increases exponentially for high-dimensional spaces.

Porta et al. (2006) and Thrun (2000) have proposed using particle filters to overcome the curse of dimensionality. Instead of dividing the state and observation spaces to a discrete grid, the beliefs are represented as a set of particles, which are then propagated with Monte-carlo methods. The complexity still depends on the number of particles chosen.

Parametric POMDPs provide another possible solution. The belief space is defined to be the parametric space of a selected function and has a fixed dimensionality equal to the number of sufficient statistics. This requires the assumption that belief propagation changes only the parameters assosiated with the chosen function, and does not change the functional form itself. Compared to methods based on discretizing the state space, the dimensionality of the belief space is reduced dramatically. With a suitable selection of the parametric form, the belief update of Equation 2.20 can be solved efficiently in closed form without iterating over the state space.

Gaussian distributions are often used in robotic localization problems and e.g. Brooks (2007), Lauri and Ritala (2014), and van den Berg, Patil, and Alterovitz (2012) propose parametric POMDP solutions based on Gaussians. If the initial belief is Gaussian and the system and observation dynamics are linear, the belief remains Gaussian. However, Gaussian distributions pose some limitations such as the inability to represent multi-modal beliefs. Moreover, van den Berg, Patil, and Alterovitz's solution does not work in domains with abrupt boundaries in sensing, e.g. inside or outside the field of view of a camera.

Porta et al. (2006) proposes a solution that handles also multi-modal beliefs by using Gaussian mixture models (GMMs). In addition to parametric representation of beliefs, the solution is extended to continuous actions and observations by using sampling.

2.2.6 Online vs. offline methods

In the methods described in the above sections, the policy is constructed offline before the execution. Offline algorithms have to calculate a policy defining which action to take in any single belief state, and thus calculating the complete policy takes a considerable amount of time for larger problems. Online algorithms on the other hand try to find an optimal policy for the current belief state only. Instead of backwards iteration, online planners use forward search from the current belief to a

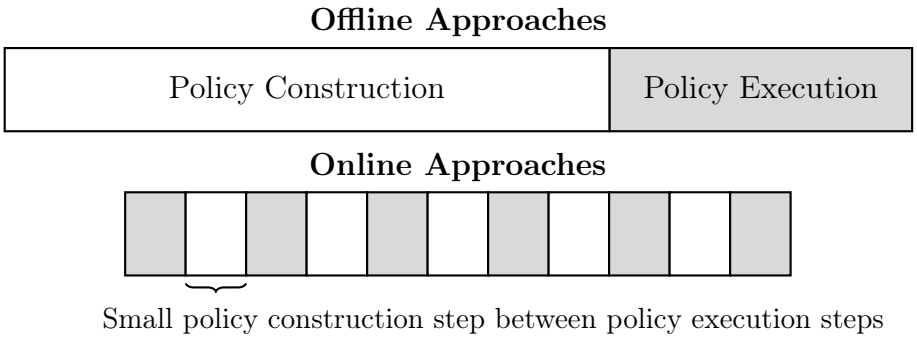


Figure 2.4 *Comparison between offline and online POMDP approaches. Figure adapted from Ross et al. (2008).*

specified depth. The advantage is that only beliefs reachable from the current belief have to be considered.

Since online planning is done at every step, there is no need to calculate the full optimal α-vector, only the maximal value for the current belief state. The policy construction and execution are interleaved as shown in Figure 2.4. Overall the policy construction and execution time is shorter for online approaches (Koenig 2001).

Ross et al. (2008) provide an excellent review of online methods. For typical online algorithm, at each timestep a single policy tree is constructed for the current belief state, where the possible sequences of actions and observations define the branches. The value of the current belief is then estimated by propagating value estimates back from the branches to their ancestors. An approximate value function computed offline can be used to estimate long-term value. After the planning phase, the execution phase executes the best action found for the current belief, after which the policy tree and belief is updated according to the observation received.

Often full-width computation that takes all possible actions and observations into account is used. The branching factor for full-width computation is $|Z| \times |A|$. This can be effective for small POMDPs, but if the action and observation spaces are large, full computation can take too long to be calculated online with any real-time constraints. Branch-and-Bound pruning can be used to stop expansion of sub-optimal branches. By keeping lower and upper bounds of a value function calculated offline, sub-optimal branches of the policy tree can be pruned so that they are never expanded and the branching factor is reduced.

Monte-Carlo sampling is another method that can be used to lower complexity when there is a large observation space. Instead of expanding the tree fully over a large set of observations, it is possible to sample a subset of the observations and consider only beliefs reached by these sampled observations. Silver and Veness (2010) have taken this further, and present an algorithm where Monte-Carlo methods are used both for belief representation and online planning.

2.3 Kalman filter

Since we consider parametric representation for the belief using Gaussian distributions, a solution for the belief propagation is provided. Kalman filtering equations (Kalman 1960) are the closed form solution to linear Gaussian Bayesian filtering problems. In this case, the purpose of Bayesian filtering is to calculate the marginal posterior distribution $b_t = p(s_t|h_t)$ of the state s_t at each time step t given the history up to that time. As stated in Section 2.2.1, the history consists of previous actions and measurements, $h_t = \{a_0, z_1, \ldots, a_{t-1}, z_t\}$.

The Bayesian filtering equations are a set of recursive equations that start with the prior distribution $b_0 = p(s_0)$. The predictive distribution at time t can be calculated with the Chapman–Kolmogorov equation

$$p(s_t|z_{1:t-1}, a_{0:t-1}) = \int_{s_{t-1} \in S} p(s_t|s_{t-1}) p(s_{t-1}|z_{1:t-1}, a_{0:t-1}) \mathrm{d}s_{t-1}. \tag{2.22}$$

Given the measurement z_t, the posterior distribution can be calculated with Bayes' rule

$$p(s_t|z_{1:t}, a_{0:t-1}) = \frac{1}{c} p(z_t|s_t, a_{0:t-1}) p(s_t|z_{1:t-1}, a_{0:t-1}), \tag{2.23}$$

where the normalization constant c is given by

$$c = \int_{s_t \in S} p(z_t|s_t, a_{0:t-1}) p(s_t|z_{1:t-1}, a_{0:t-1}) \mathrm{d}s_t. \tag{2.24}$$

This is of the same form as the belief update function in Equation 2.20.

Let us define the linear Gaussian dynamic and measurement models as

$$s_t = F_{t-1} s_{t-1} + q_{t-1}, \tag{2.25a}$$

$$z_t = M_t s_t + w_t, \tag{2.25b}$$

where $q_{t-1} \sim \mathcal{N}(0, Q)$ is the process noise, and $w_t \sim \mathcal{N}(0, W)$ is the measurement noise, the matrix F_{t-1} is the state transition matrix, and M_t the measurement model matrix. F_{t-1} and M_t may depend on the action a_{t-1}.

The Bayesian filtering equations (2.22, 2.23) for model 2.25ab can be solved in closed form, so that they remain Gaussian. The resulting distributions are:

$$p(s_t|z_{1:t-1}, a_{0:t-1}) = \mathcal{N}(\mu_t^+, \Sigma_t^+), \tag{2.26a}$$

$$p(s_t|z_{1:t}, a_{0:t-1}) = \mathcal{N}(\mu_t, \Sigma_t). \tag{2.26b}$$

The mean and covariance for the Gaussian distribution in Equation 2.26a are given by the Kalman filter prediction step:

$$\mu_t^+ = A_{t-1}\mu_{t-1}, \tag{2.27a}$$
$$\Sigma_t^+ = A_{t-1}\Sigma_{t-1}A_{t-1}^\top + Q, \tag{2.27b}$$

and the ones in Equation 2.26b by the Kalman filter update step:

$$d_t = z_t - H_t\mu_t^+) \tag{2.28a}$$
$$E_t = H_t\Sigma_t^+ H_t^\top + W \tag{2.28b}$$
$$K_t = \Sigma_t^+ H_t^\top E_t^{-1} \tag{2.28c}$$
$$\mu_t = \mu_t^+ + K_t d_t \tag{2.28d}$$
$$\Sigma_t = \Sigma_t^+ - K_t E_t K_t^\top, \tag{2.28e}$$

The recursion is started with the prior information of state $s_0 \sim \mathcal{N}(\mu_0, \Sigma_0)$.

Proof of Bayesian filtering and Kalman filter equations is bypassed and is presented e.g. by Särkkä (2013, pp. 56–58). However, the original Kalman filter can only handle linear models.

2.4 Extended Kalman filter

The extended Kalman filter (EKF) is an extension of the linear Kalman filter, and is based on the linearization of the dynamic and measurement function to the expected state. The general non-linear dynamic and measurement models, not limited to additional noise, can be written as

$$s_t = f(s_{t-1}, a_{t-1}, q_{t-1}), \tag{2.29a}$$
$$z_t = m(s_t, a_{t-1}, w_t), \tag{2.29b}$$

where $q_{t-1} \sim \mathcal{N}(0, Q)$ is the process noise, and $w_t \sim \mathcal{N}(0, W)$ is the measurement noise, a_{t-1} is the action taken at time step $t-1$, $f(\cdot)$ is the dynamic model function, and $m(\cdot)$ is the measurement model function.

EKF is based on Taylor series approximation of the non-linearities. For first order approximation, the EKF prediction step becomes:

$$\mu_t^+ = f(\mu_{t-1}, a_{t-1}, 0) \tag{2.30a}$$
$$\Sigma_t^+ = F_s\Sigma_{t-1}F_s^\top + F_q Q F_q^\top, \tag{2.30b}$$

where F_s and F_q are Jacobians of the dynamic model function f, with respect to state and noise respectfully, evaluated at μ_{t-1}.

The corresponding update step is:

$$d_t = z_t - m(\mu_t^{i+}, a_{t-1}, 0) \tag{2.31a}$$
$$E_t = M_s \Sigma_t^{i+} M_s^\top + M_w W M_w^\top \tag{2.31b}$$
$$K_t = \Sigma_t^{i+} M_s^\top E_t^{-1} \tag{2.31c}$$
$$\mu_t^i = \mu_t^{i+} + K_t d_t \tag{2.31d}$$
$$\Sigma_t^i = \Sigma_t^{i+} - K_t E_t K_t^\top, \tag{2.31e}$$

where M_s and M_w are Jacobians of the measurement model function m, with respect to state and noise respectfully, evaluated at μ_t^+. Proof is omitted and can be found in Särkkä (2013, pp. 69–72).

3. PROBLEM FORMULATION

3.1 Case description

A four-wheeled mobile robot has three machine vision cameras mounted on it. The direction of gaze of each camera can be controlled independently of the robot pose. The robot has to select the direction of gaze of its vision system while following a given sequence of landmarks. The a priori information of each landmark position is a highly uncertain multivariate Gaussian distribution. From navigational point of view, the most important things to consider are 1) to which landmark is the robot heading at the moment, and 2) to which landmark should it head next.

To form accurate enough landmark position estimates for navigation, the task is formulated as an information-gathering problem. The problem can be divided into two separate subtasks: 1) maximizing information about the position of the current target landmark in sequence, and 2) maximizing information about the position of the next target landmark in sequence. The robot moves at a constant velocity and both subtasks are solved to maximize the probability that the robot actually visits all of the landmarks. When the robot believes it is close enough to the current landmark, it starts navigating to the next.

The three cameras on the robot can be utilized in the following configuration: (see Figure 1.1) (a) all cameras face forward, (b) left and middle cameras face forward while the right camera faces right, (c) right and middle cameras face forward while the left camera faces left, (d) left and middle cameras face left while the right camera faces forward, (e) right and middle cameras face right while the left camera faces forward, or (f) all cameras face different directions. When any two cameras face the same direction, they are used as a stereo pair.

3.2 Dynamics and observation models

The robot's state s_t^r at time instant t is given by its location x_t, y_t, and its heading θ_t. The robot is controlled by its translational velocity v_t, and its rotational velocity

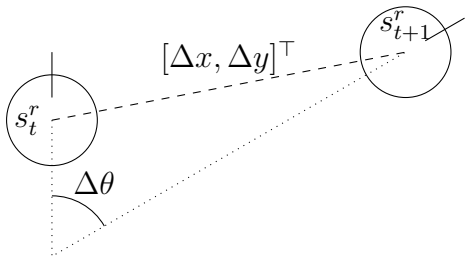


Figure 3.1 *Illustration of robot movement between time instances t and $t+1$.*

ω_t. The robot's relative movement between time instances t and $t+1$, as depicted in Figure 3.1, is calculated from the control signals as follows

$$\begin{bmatrix} \Delta x \\ \Delta y \\ \Delta\theta \end{bmatrix} = \begin{bmatrix} v_t \Delta t \cos(\omega_t \Delta t) \\ -v_t \Delta t \sin(\omega_t \Delta t) \\ \omega_t \Delta t \end{bmatrix}. \tag{3.1}$$

The control is affected by additive Gaussian noise, so that the joint probability density function (pdf) of $u_t = [v_t, \omega_t]^\top$ is $\mathcal{N}(\hat{u}_t, Q)$, where $\hat{u}_t$ are the desired control inputs and $Q = \mathrm{diag}(\sigma_v^2, \sigma_\omega^2)$ is the noise covariance. The control at each time step is presented in Section 3.4.

The N stationary landmarks are positioned at locations $l^i = \{l_i^x, l_i^y\}_{i=1}^N$ and the system state is defined as the coordinates of each landmark in the robot's coordinate frame, $s_t = [l_t^1, l_t^2, \ldots, l_t^N]^\top$. The landmark are fully distinguishable, but the sizes of the landmarks are unknown. The transition model $l_{t+1}^i = f(l_t^i, u_t)$ for each landmark can be written as

$$l_{t+1}^i = \begin{bmatrix} \cos(\Delta\theta) & \sin(\Delta\theta) \\ -\sin(\Delta\theta) & \cos(\Delta\theta) \end{bmatrix} l_t^i - \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}. \tag{3.2}$$

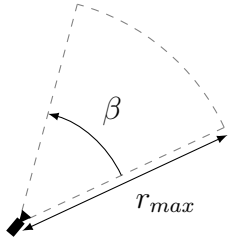


Figure 3.2 *Illustration of the camera cone of observation $C(a_t, \alpha, r_{max})$ for a single camera. The action a_t affects the gaze direction i.e. orientation of the camera.*

The robot can observe landmarks that reside inside an area visible to the cameras $VA(C_{j=1:3})$. The visible area is determined by the cameras' cone of observations $C_j(a_t, \beta, r_{max})_{j=1:3}$ which are defined by the action $a_t \in A$, view angle β, and maximum range r_{max}. The cone of observation of a camera can be seen in Figure 3.2

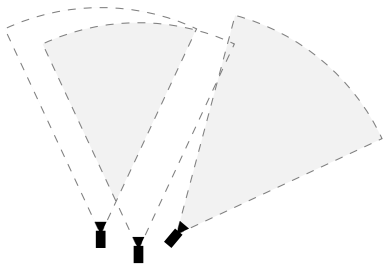


Figure 3.3 *Illustration of the area from which we get measurements (gray) in one of the camera modes. The left and middle cameras are operated as a stereo pair i.e. we get measurements only from the area visible to both cameras, whereas the right camera is operated in monocular mode.*

and the formation of the visible area in Figure 3.3. The action space A is the six camera modes presented in Figure 1.1, and the current action determines which cameras are operated in stereo and which in monocular mode.

In stereo mode the observation consists of the measured landmark coordinates l_t^i and the measurement model $z_t = m_{stereo}(l_t^i) + w_t^{stereo}$ can be written

$$z_t = l_t^i + w_t^{stereo}, \tag{3.3}$$

where w_t^{stereo} is Gaussian noise with zero mean and covariance $W^{stereo} = \text{diag}(\sigma_x^2, \sigma_y^2)$. The measured landmark coordinates are calculated via linear triangulation from the coordinates of the landmark in the camera coordinate frames (Hartley and Zisserman 2004, pp. 310–313), but only the x and y -coordinates are taken into account in the robot coordinate frame, and the z-coordinate of the landmark is ignored. The projection matrices for all the cameras in all different camera modes are calculated a priori and are known.

In monocular mode only the angle δ of the landmark relative to the robot's coordinate frame can be measured and the measurement model $z_t = m_{mono}(l_t^i) + w_t^{mono}$ is

$$z_t = \delta_t + w_t^{mono} = \arctan\left(\frac{l_t^{i,y}}{l_t^{i,x}}\right) + w_t^{mono}, \tag{3.4}$$

where w_t^{mono} is Gaussian noise with zero mean and variance $W^{mono} = \sigma_\delta^2$. The measured angle is the arctangent of the deviance of the landmark's x-coordinate from the center of the camera coordinate frame divided by the focal length of the camera. The structure of the measurement system in question, and the forming of the output measurement are more thoroughly explained by Välimäki (2015).

3.3 State estimation

The state space is continuous and unbound, which results in an infinite-dimensional belief space if defined as a pdf over the state space. Instead, we use a parametric representation of the belief space and assume that the belief can be approximated as a Gaussian distribution. This way, the belief propagation can be done directly in the low-dimensional parametric space.

We apply an EKF, as presented in Section 2.4, to track the robot's belief about the landmark locations, $L_t^i|_{i=1:N}$, which are assumed to be independent random variables. Given the previous belief $L_{t-1}^i \sim b_{t-1}^i = \mathcal{N}(\mu_{t-1}^i, \Sigma_{t-1}^i)$, at each timestep we first calculate the prediction on the following time step $L_t^{i+} \sim b_t^{i+} = \mathcal{N}(\mu_t^{i+}, \Sigma_t^{i+})$ according to

$$\mu_t^{i+} = f(\mu_{t-1}^i, \hat{u}_{t-1}) \tag{3.5a}$$
$$\Sigma_t^{i+} = F_s \Sigma_{t-1}^i F_s^\top + F_q Q F_q^\top, \tag{3.5b}$$

where F_s and F_q are Jacobians of the transition function f with respect to state and control noise respectfully, evaluated at $(\mu_{t-1}^i, \hat{u}_{t-1})$:

$$\begin{aligned} F_s &= \left.\frac{\mathrm{d}f(l^i, u)}{\mathrm{d}l^i}\right|_{l^i=\mu_{t-1}^i, u=\hat{u}_{t-1}} \\ &= \left.\begin{bmatrix} \cos(\omega\Delta t) & \sin(\omega\Delta t) \\ -\sin(\omega\Delta t) & \cos(\omega\Delta t) \end{bmatrix}\right|_{l^i=\mu_{t-1}^i, u=\hat{u}_{t-1}} \\ F_q &= \left.\frac{\mathrm{d}f(l^i, u)}{\mathrm{d}u}\right|_{l^i=\mu_{t-1}^i, u=\hat{u}_{t-1}} \\ &= \left.\begin{bmatrix} -\Delta t\cos(\omega\Delta t) & \Delta t(v\Delta t\sin(\omega\Delta t) - l_i^x\sin(\omega\Delta t) + l_i^y\cos(\omega\Delta t)) \\ \Delta t\sin(\omega\Delta t) & \Delta t(v\Delta t\cos(\omega\Delta t) - l_i^x\cos(\omega\Delta t) - l_i^y\sin(\omega\Delta t)) \end{bmatrix}\right|_{l^i=\mu_{t-1}^i, u=\hat{u}_{t-1}} \end{aligned}$$

Every time a measurement is received, the posterior $L_t^i|a_{t-1} \sim b_t^i = \mathcal{N}(\mu_t^i, \Sigma_t^i)$ is calculated according to

$$d_t = z_t - m(\mu_t^{i+}) \tag{3.6a}$$
$$E_t = M_s \Sigma_t^{i+} M_s^\top + W \tag{3.6b}$$
$$K_t = \Sigma_t^{i+} M_s^\top E_t^{-1} \tag{3.6c}$$
$$\mu_t^i = \mu_t^{i+} + K_t d_t \tag{3.6d}$$
$$\Sigma_t^i = \Sigma_t^{i+} - K_t E_t K_t^\top. \tag{3.6e}$$

Depending which measurement is received, the appropriate measurement function

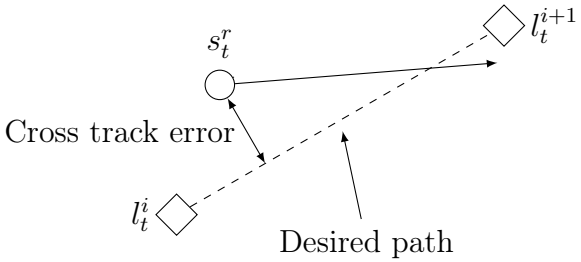


Figure 3.4 *Illustration of cross track error. Cross track error is the perpendicular distance of the robot from the desired path. The bigger the cross track error, the more aggressively the robot is steered towards the desired path.*

m_{stereo} or m_{mono} is used, and M_s is the Jacobian of the measurement function with respect to state:

$$M_s = \begin{cases} I & \text{if stereo measurement} \\ \begin{bmatrix} \frac{-l_i^y}{l_i^{x\,2}+l_i^{y\,2}} & \frac{l_i^x}{l_i^{x\,2}+l_i^{y\,2}} \end{bmatrix} & \text{if monocular measurement,} \end{cases}$$

where I is the identity matrix.

The state development in the prediction step is independent of the action, whereas the update step represents the belief update function $\tau(b, a, z')$.

3.4 Robot control

As mentioned above in Section 3.2, the robot is controlled by its translational velocity v_t, and its rotational velocity ω_t. To keep the focus of this thesis on the optimization of the gaze direction of the camera system, we use only a simple control scheme for the robot: The robot is set to move forward with a constant translational velocity, $v_t = v$, until it believes to be close enough to the last landmark in sequence.

The rotational velocity for each time step, ω_t, is calculated with a discrete time proportional-integral-derivative (PID) controller of the form

$$\omega_t = K_p e_t + K_i \sum_{i=1}^{t} e_i \Delta t_i + K_d \frac{e_t - e_{t-1}}{\Delta t}, \tag{3.7}$$

where K_p, K_i, and K_d are parameters of the PID controller, e_t is a cross track error as illustrated in Figure 3.4, and Δt is the time difference between two consecutive control steps. Cross track error measures the perpendicular distance of the robot from the desired path. In order to achieve smooth paths, the cross track error is calculated for a point just in front of the robot.

The desired path consists of linear line segments starting from the robot's current position to the mean of the pdf of the next landmark in sequence, and then going from landmark to landmark until the end of sequence. A new path is calculated every time a new prediction or update is available from the state estimation equations 3.5 and 3.6.

The navigation is based purely on the robot's belief of the landmark positions with respect to its own position, and therefore the robot might believe to have reached the end of the sequence even if it has not observed any of the landmarks, i.e. the robot could just navigate to the landmark locations in the initial belief. A more rational navigation strategy could be to stop if the robot believes to have reached a landmark, but no measurements have been obtained, to allow time to observe the surroundings. Another possibility could be to circle the area with highest probability of finding the landmark.

3.5 Reward function

As the task consists of gaining information about the current and next landmarks in sequence, we need a way to measure the expected information gain. Information theoretic quantities such as the Kullback-Leibler (KL) divergence (Kullback 1968) are often used to measure the information gain between two pdfs. The expected KL divergence can be expressed as the mutual information (MI) of the random variables.

We want to maximize the MI, $\mathcal{I}(L_{t+1}^{i+}, L_{t+1}^{i}|a_t)$, of the predicted and expected landmark locations, so that the total immediate reward is

$$\begin{aligned}\rho(b_t, a_t) = \vartheta\mathcal{I}(L_{t+1}^{i+}, L_{t+1}^{i}|a_t) \\ + \varphi\mathcal{I}(L_{t+1}^{i+1+}, L_{t+1}^{i+1}|a_t) - g(a_t, a_{t-1}),\end{aligned} \tag{3.8}$$

where ϑ and φ are weighing factors to prioritize gathering information either about the current or about the next landmark and $g(a_t, a_{t-1})$ is a cost for changing camera modes defined as

$$g(a_t, a_{t-1}) = \begin{cases} \kappa, & a_t \neq a_{t-1} \\ 0, & a_t = a_{t-1},\ t = 0 \end{cases}, \tag{3.9}$$

where κ is a positive scalar constant.

MI of the predictive and expected landmark locations can be calculated as

$$\begin{aligned}\mathcal{I}(L_{t+1}^{i+}, L_{t+1}^{i}|a_t) &= \mathbb{E}_Z\left[D_{KL}(b_{t+1}^{i+}, b_{t+1}^{i})\right] \\ &= \int_{z_{t+1}\in Z} D_{KL}(b_{t+1}^{i+}, b_{t+1}^{i}) p(z_{t+1}|a_t, b_t^i)\mathrm{d}z_{t+1}\end{aligned} \tag{3.10}$$

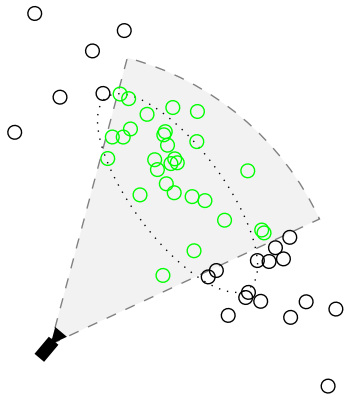


Figure 3.5 *Illustration of the Monte-Carlo sampling. The dotted ellipse represents the 50% confidence interval of the landmark pdf, L_{t+1}^{i+}, from which the K samples, denoted by circular markers, are drawn. If the k^{th} sample lies inside the visible area, we expect to get a measurement z_{t+1}^k.*

where b_{t+1}^i depends on the observation z_{t+1}, and $p(z_{t+1}|a_t, b_t^i)$ is the probability that the landmark l_{t+1}^i is visible in observation z_{t+1} if action a_t is taken in current belief b_t^i. $D_{KL}(\cdot)$ is the KL divergence. The KL divergence between two multivariate Gaussian distributions, $x_1 = \mathcal{N}(\mu_1, \Sigma_1)$ and $x_2 = \mathcal{N}(\mu_2, \Sigma_2)$, can be calculated as

$$D_{KL}(x_1, x_2) = \frac{1}{2}\left((\mu_1 - \mu_2)^\top \Sigma_1^{-1}(\mu_1 - \mu_2) + \mathrm{tr}(\Sigma_1^{-1}\Sigma_2) - d + \ln\frac{|\Sigma_1|}{|\Sigma_2|}\right), \quad (3.11)$$

where $\mathrm{tr}(\cdot)$ denotes the trace operation, $|\cdot|$ matrix determinant, $\ln(\cdot)$ the natural logarithm, and d is the dimensionality of the space where the Gaussians are defined (see e.g. Cover and Thomas 2006).

The probability of l_{t+1}^i being visible in observation z_{t+1} would be difficult to calculate, so instead we use Monte-Carlo methods and draw K samples from the probability distribution; $\{z_{t+1}^k\}_{k=1}^K \sim p(z_{t+1}|a_t, b_t^i) = \mathcal{N}(m(\mu_{t+1}^{i+}), E_{t+1})$. In other words, we sample possible landmark locations from $\mathcal{N}(\mu_{t+1}^{i+}, \Sigma_{t+1}^{i+})$ and expect to get a measurement z_{t+1}^k from each sample that lies inside the appropriate cones of observation $C_j(a_t, \alpha, r_{max})_{j=1:3}$ that form the visible area. The sampling is illustrated in Figure 3.5. The camera mode a_t determines which cameras are used as a stereo pair, i.e. the sample must be inside both cones of observation, and which as monocular cameras. Now (3.10) can be approximated with

$$\mathcal{I}(L_{t+1}^{i+}, L_{t+1}^i|a_t) \approx \frac{1}{K}\sum_{k=1}^{K} D_{KL}(b_{t+1}^{i+}, b_{t+1}^i), \quad (3.12)$$

where $b_{t+1}^i = \tau(b_{t+1}^{i+}, a_t, z_{t+1}^k)$ depends on the sampled measurement z_{t+1}^k, and further $D_{KL}(b_{t+1}^{i+}, b_{t+1}^i) = 0$ for every k when there is no measurement expected.

Although MI is a good measure of immediate information gain, long term performance is better measured with the development of differential entropy. The differential entropy $h(\cdot)$ for a multivariate Gaussian distribution is calculated as

$$h(\mathcal{N}(\mu, \Sigma)) = \frac{1}{2} \ln(2\pi e)^d |\Sigma|, \tag{3.13}$$

where e is the Euler's number and d the dimensionality of the space where the Gaussian is defined (Cover and Thomas 2006).

4. SOLUTION AND IMPLEMENTATION

4.1 Solving the problem

Standard POMDP solvers cannot be used for this problem since we use information theoretic rewards. A summary of the problem defined in the previous Chapter is provided in Table 4.1.

Table 4.1 *Summary of selected methodology.*

Aspects	**Methodology**
Belief representation	independent Gaussian distributions
Belief propagation	EKF
Robot control	PID control to minimize cross track error
Reward function	MI approximated with Monte-Carlo methods

To avoid added complexity, myopic optimization of the gaze direction was chosen, i.e. we act greedily. The planning is done online, so at each timestep t we solve

$$\pi^* = \arg\max_{a_t \in A} \rho(b_t, a_t) \tag{4.1}$$

to obtain the optimal policy, which for a myopic case is the single optimal action $\pi^* = a_t^*$. The optimal action is executed and an observation is received.

Acting greedily may result in poorer performance if 1) the focus of attention cannot be changed rapidly, or 2) the observed features are not stationary (Lauri and Ritala 2014). In the case covered by this thesis, the landmarks are defined to be stationary, however, the penalty set for changing camera modes controls how rapidly the focus of attention can be changed. In a myopic solution the utility of actions is measured only one step ahead, i.e. the expected immediate reward for changing camera modes has to be larger than the penalty assosiated with it, or the camera mode is not changed.

4.2 Implementation

The algorithms described in the previous chapter were implemented on ROS (Quigley et al. 2009) using C++ and Python. The source files comprise approximately 3300 lines of code. The implemented software can be divided into four main components as shown in Figure 4.1: 1) perception, 2) state estimation, 3) action selection, and 4) path following.

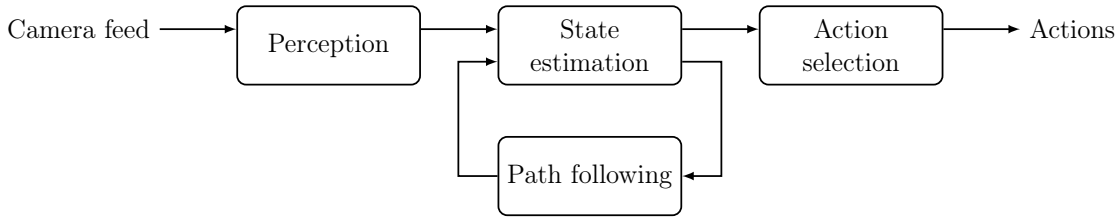


Figure 4.1 *The main components of the implemented system.*

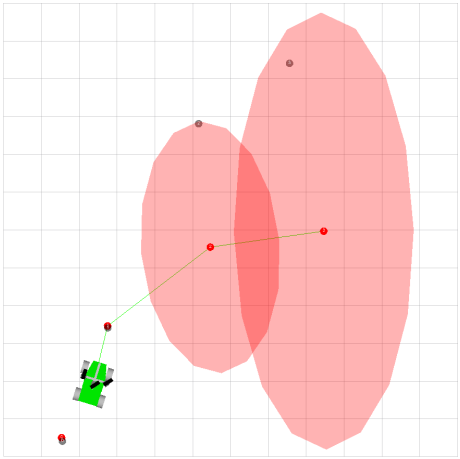


Figure 4.2 *Screen capture of the visualization during simulation. Red cones denote robot's belief of landmark locations, with the assosiated ellipses representing 50% confidence intervals, and the green line denotes the planned trajectory. The translucent black cones denote the actual landmark positions, unknown to the robot.*

Perception receives the image feed from the cameras, detects possible landmarks in the images, and outputs the measurements. Quick response (QR) codes are used as landmarks since they are easy to recognize from the images and identify. The ROS package `hector_qrcode_detection`[1], developed by Team Hector of Technische Universität Darmstadt, is used to find and decode the QR codes. Some changes were needed to adapt the package to the newer ROS version used.

State estimation implements the algorithms introduced in Section 3.3. A new prediction is made every time a new control signal is received from the path following module, and the state estimate is updated with every measurement received from

[1] `http://wiki.ros.org/hector_qrcode_detection`, 29th October, 2015

Algorithm 4.1 Action selection algorithm

Require: N is the total number of landmarks, i is the current landmark in sequence, b^+ is the belief after an EKF prediction step, and a' is the previous action

procedure ActionSelection(b^+)
  for each action $a \in A$ **do**
    if $i \leq N - 1$ **then**
      $\rho(b, a) = \vartheta\mathcal{I}(L^{i+}, L^i|a) + \varphi\mathcal{I}(L^{i+1+}, L^{i+1}|a) - g(a, a')$
    else if $i = N$ **then**
      $\rho(b, a) = \vartheta\mathcal{I}(L^{i+}, L^i|a) - g(a, a')$
    else
      return do nothing
    end if
  end for
  $a^* = \arg\max_{a \in A} \rho(b, a)$
  return do a^*
end procedure

function $\mathcal{I}(L^{i+}, L^i|a)$
  $MI = 0$
  draw K samples from b^{i+}
  for each sample $l_k^{i+}|_{k=1:K}$ **do**
    if $l_k^{i+} \in VA(C_{j=1:3})$ **then**
      $z^k = m(l_k^{i+})$
      $b^i = \text{EkfUpdate}(b^{i+}, z^k)$ ▷ Equation 3.6
    end if
    $MI \mathrel{+}= D_{KL}(b^{i+}, b^i)$ ▷ Equation 3.11
  end for
  return MI/K
end function

the perception module. The state estimation module outputs the predictions to the action selection module at a reduced rate, and calculates a new path for the path following module with every prediction or update.

Action selection calculates the expected rewards for all of the actions and selects the one that yields the best immediate reward. A more detailed description of the action selection precedure is presented in Algorithm 4.1. The action selection procedure is run every time there is a new state prediction available from the state estimation module.

Path following receives a new path from the state estimation module with every new estimate and calculates new control signals as stated in Section 3.4. In additions to the motor controllers of the robot, the control signals are sent to the state estimation module.

Furthermore, a model of the robot (seen later in Figure 6.1) was created for visu-

alization purposes, and the program can be visualized online using `rviz`. A screen capture of the visualization during one of the simulation experiments described in the next Chapter can be seen in Figure 4.2.

5. SIMULATION EXPERIMENTS

5.1 Simulation environment

The benefits of using the active sensing implementation described in the previous sections are demonstrated with simulation experiments. A comparison is made between the dynamic camera configuration of the active sensing and fixed front-facing stereo cameras, i.e. the cameras are permanently in camera mode (a) of Figure 1.1. Both methods were tested in two scenarios with different prior information. The experiment was repeated 10 times for each method and scenario.

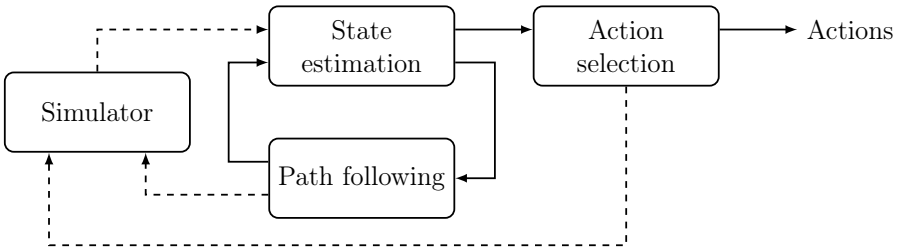
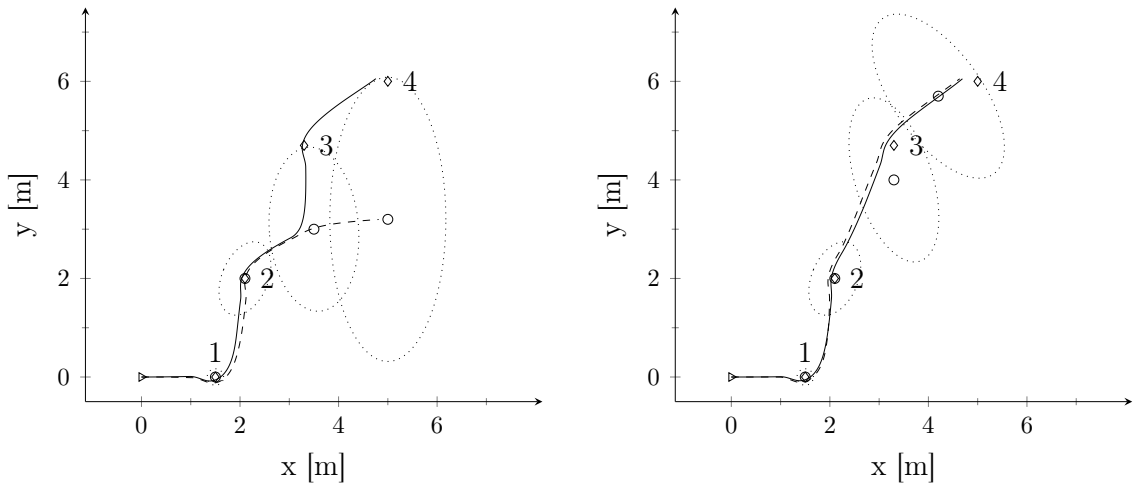


Figure 5.1 *The relation of the simulator to the main program components.*

In addition to the main software components described in Chapter 4.2, a simulator was developed to replace the perception module. The relation of the simulator to the other components is visible from Figure 5.1. The simulator advances the state according to the state transition model of Equation 3.2, and produces measurements, according to measurement functions in Equations 3.3 and 3.4, for the state estimation module. The measurements depend on the current action received from the action selection module. Noise is added both to the state transition and measurements.

The sequence of landmarks in the simulation environment consisted of $N = 4$ landmarks. The robot moves at a constant velocity of 0.3 m/s and starts navigating to the next landmark when it believes to be closer than 0.2 m to the current landmark in sequence. The state estimation module outputs predictions to the action selection module at 1 Hz rate. Different initial beliefs of the landmark positions in the two tested scenarios with 50% confidence intervals can be seen in Figure 5.2 with the actual landmark positions, and the initial position of the robot.



(a) *Hight element of surprise.* (b) *Low element of surprise*

Figure 5.2 *Simulation experiments both with and without using the active sensing. The initial position of the robot is marked with the triangle. Circular markers denote the prior of the landmarks, with the assosiated dotted ellipses representing the 50% confidence intervals. Diamond markers denote the actual positions of the landmarks. The dashed line is the robot trajectory when active sensing is not used and the solid line is the robot trajectory when using the proposed active sensing.*

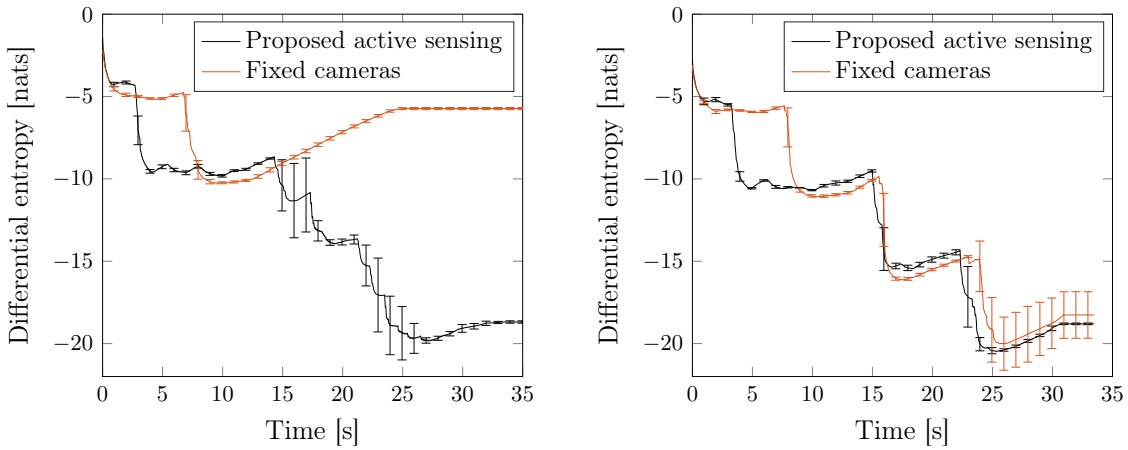
Let us construct a notion of element of surprise, where the element of surprise indicates the distance between the actual landmark position and the position of the prior mean. If the distance between the two positions is large, the element of surprise is high, and vice versa. One of the tested scenarios represents high element of surprise, whereas the other represents low element of surprise. With this notion, the width of the belief pdf translates to how likely a surprising position is.

The gaze direction of the cameras is $40°$ when looking left, $-40°$ when looking right, and $0°$ when facing forward. The angle of view of the cameras is $\beta = 50°$ and maximum range $r_{max} = 2.5$ m. The noise parameters used were $\sigma_v^2 = \sigma_\omega^2 = 0.01$ m^2/s^2 and rad^2/s^2 accordingly, $\sigma_x^2 = 0.01$ m^2, $\sigma_y^2 = 0.02$ m^2, and $\sigma_\delta^2 = 0.05$ rad^2. The weighing factors were $\vartheta = 0.7$, $\varphi = 0.3$, and the penalty $\kappa = 0.01$.

The effect of the number of samples to draw for the Monte-Carlo sampling was also tested with simulations when $K \in \{10, 50, 100, 500, 5000\}$. The comparison with the fixed front facing stereo cameras was performed with $K = 10$.

5.2 Results

Figure 5.2 presents typical outcomes of the simulations in both tested scenarios. In the case of Figure 5.2a the robot with fixed cameras never reaches all of the landmarks. It fails to see the actual third landmark, with high element of surprise,



(a) *Prior distribution as depicted in Figure 5.2a.*

(b) *Prior distribution as depicted in Figure 5.2b.*

Figure 5.3 *Time evolution of belief state entropy. Lower is better. The lines indicate mean differential entropy over 10 experiments and the bars indicate 95% confidence intervals.*

situated to the left of its originally planned trajectory, and continues towards the prior mean position. When the proposed active sensing is used, the robot keeps observing its surroundings in order to minimize the uncertainty about the landmark and upon observing the actual landmark, corrects its belief and plans a new trajectory to follow.

To quantify the results, the differential entropy of the belief state during the experiments was studied and can be seen in Figure 5.3a. Lower entropy indicates lower uncertainty. Using the proposed active sensing, the robot visited all the landmarks on each experiment, whereas the robot with fixed cameras never observed the third and fourth landmarks and only navigated to the prior mean locations. This can be verified from the development of differential entropy; after the second sharp decline, where the robot first observes a new landmark, the differential entropy starts increasing due to the control noise, and finally reaches a steady state when the robot stops. With the proposed active sensing, the differential entropy keeps declining as new landmarks are observed, and a much lower final value is reached.

If, however, the actual landmark positions are close to the prior means, i.e. element of surprise is low, there is little difference between the performance of the proposed active sensing and fixed cameras as seen in Figure 5.2b. From the sharp declines of differential entropy of the belief state in Figure 5.3b, we can see that the active sensing method intuitively observes the landmarks sooner than the method with fixed front-facing cameras. The fixed camera method, however, reaches lower differential entropy values after observing a landmark, because all three cameras are used as stereo pairs, whereas the active sensing methods usually only observes the landmark

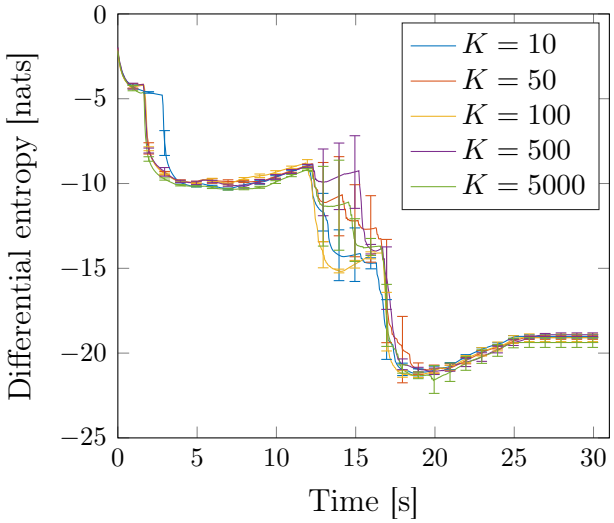


Figure 5.4 *Time evolution of belief state entropy with different sample sizes K. Lower is better. The lines indicate mean differential entropy over 10 experiments and the bars indicate 95% confidence intervals.*

with one stereo pair, and operates the third camera in a monocular mode. There is no significant difference in the final entropy of the two methods. The fixed camera method failed to observe the fourth landmark in one of the experiments, which resulted in the large confidence intervals seen at the end.

The effect of the number of samples drawn for the Monte-Carlo sampling in the reward estimation was also studied in a similar setting as that of Figure 5.2a. As can be seen from Figure 5.4 the difference in the overall performance, if measured by the differential entropy, is not significant. However, larger sample sizes, $K > 100$, tend to prefer a camera mode where multiple cameras face toward the prior mean position of the landmark, even when the information is highly uncertain. This resulted to planners with larger sample sizes to observe the third landmark somewhat later than the planners with smaller sample sizes.

This behavior is likely caused by the chosen reward function. Observing the landmark with a pair of stereo cameras always reduces the entropy more than observing it with a monocular camera. With large sample sizes, the area closest to the prior mean is most densely populated, and if one or multiple stereo pairs are directed toward it, the expected reward is at its highest. In another scenario than the one tested, this could also potentially lead to unobserved landmarks.

Using too small a sample size, that does not sufficiently represent the pdf being sampled, results in poor optimization of the gaze direction and erratic behavior. This is the case with $K = 10$ samples, as can be seen from Figure 5.5, which compares the time evolution of gaze direction of the three cameras during a single simulation with sample sizes $K \in \{10, 100, 5000\}$. The camera mode is frequently changed with

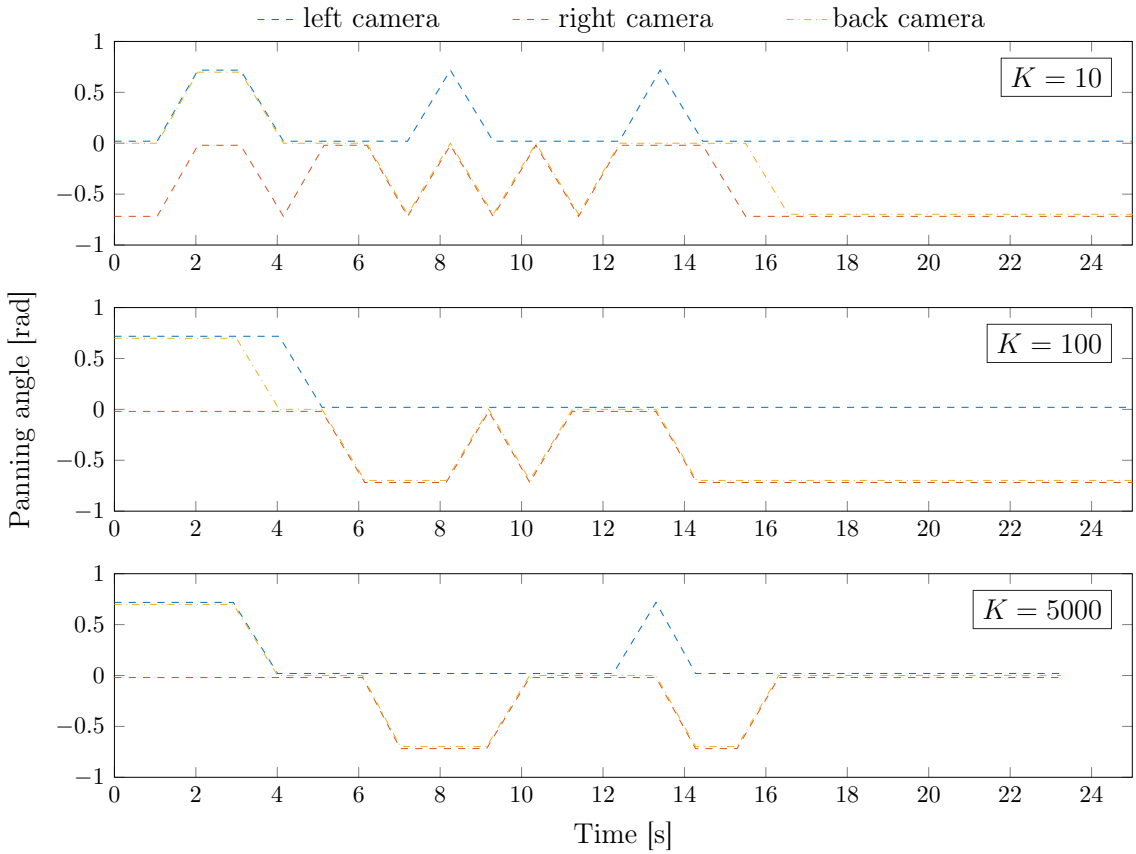


Figure 5.5 *Time evolution of gaze direction of the three cameras during one simulation with different sample sizes K.*

$K = 10$, whereas $K = 100$ and $K = 5000$ show a more composed behavior.

Based on the simulations, using the proposed active sensing is especially beneficial when the prior information about the landmark positions is highly uncertain and the actual landmark positions differ from the prior means, i.e. element of surprise is high and a surprise is likely to happen. If prior information is accurate (surprise unlikely), or conversely the prior mean happens to be close to the actual landmark location (low element of surprise), there is no significant difference between using fixed front facing cameras or the dynamic active sensing solution.

This is due to fixed cameras being unable to cope with surprising landmark positions. The robot is heading towards the prior mean position, therefore the cameras are directed towards the prior mean. If the element of surprise is larger than the field of view of the cameras, the landmark is unlikely to be observed. With active sensing the observable area is larger, and landmarks with high element of surprise are observed with greater likelihood.

Drawing $K = 100$ samples for the estimation of MI provided the best results, but as the simulations were repeated only 10 times and were not extensive, no definite conclusion about the best number of samples to use can be made.

6. FUTURE REAL-WORLD EXPERIMENTS

Experiments done on a four wheeled mobile robot seen in Figure 6.1 have been planned. The robot has a 3.1 GHz Intel i7 processor, 16 GB of RAM and runs ROS on Linux. Three Point Grey Grasshopper3 USB3 machine vision cameras with 2048×2048 resolution are mounted to the robot. In addition that the panning angle of all the cameras can be controlled between $-90°$ and $90°$, the tilt of the front cameras can also be controlled. This, however, is not utilized in the presented case study. A more detailed description of hardware is provided in Table 6.1.

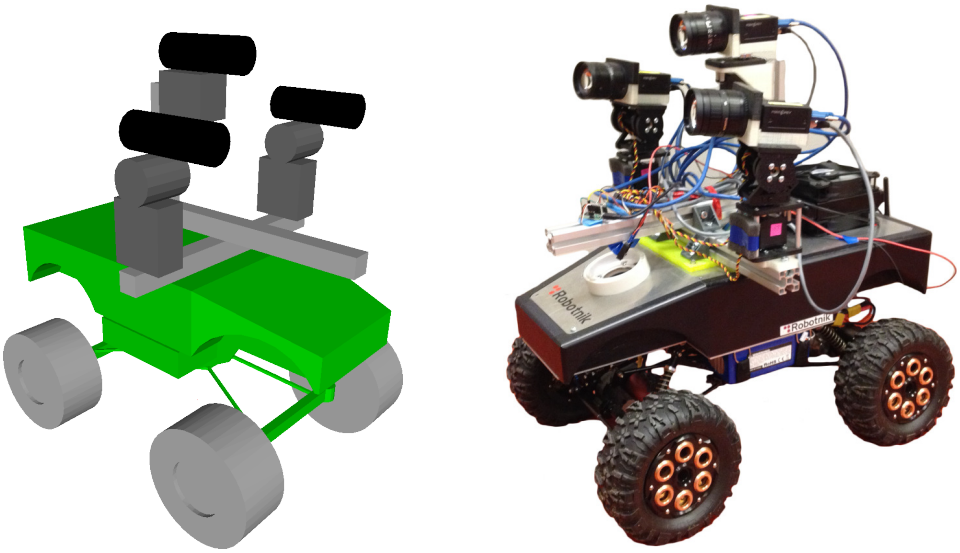


Figure 6.1 *The four wheeled mobile robot used to carry out future experiments. along with a model created for simulation and visualization purposes.*

Välimäki (2015) describes the development of the camera rig control and operation. The results discussed therein included problems with image acquisition, frame rate, and reduced resolution. All of these were resolved by installing an additional USB3 host controller.

Some experiments with the robot were conducted, but the dynamics model of Equation 3.2 has proven to be too inaccurate to cope with real world situations. The robot features soft rubber tires and the control velocities poorly translate to actual movement velocities. As the robot currently provides no other means to track its pose and relative movement, navigating over even short distances is problematic. An

Table 6.1 *Hardware specifications. Table adapted from Välimäki (2015).*

Component	**Specification**
Motherboard	Gigabyte GA-Q87TN[a]
Processor	Intel i7 3770S[b]
RAM	16GB
Host adapter	Fresco FL1100, 4 port USB 3.0 Host Controller Card[c]
Camera	3 × Point Grey Grasshopper3 GS3-U3-41C6C-C[d]
	2048 × 2048, 90 FPS
	CMOSIS CMV4000-3E5 CMOS
Lens	3 × Fujinon CF12.5HA-1[e]
	Focal length: 12.5mm
	Iris range: F1.4∼F22
	Angle of view: 45°13' × 42°01'
Operating system	Ubuntu 14.04 LTS
	ROS Indigo Igloo[f]

[a] `www.gigabyte.us/products/product-page.aspx?pid=4755\#ov`
[b] `ark.intel.com/products/65524`
[c] `www.ptgrey.com/usb-3-pcie-20-x1-host-adapter-card-4-port`
[d] `www.ptgrey.com/grasshopper3-41-mp-color-usb3-vision-cmosis-cmv4000-2-camera`
[e] `www.fujifilm.eu/eu/products/optical-devices/cctv-and-machine-vision/p/cf125ha-1`
[f] `wiki.ros.org/indigo`

inertial measurement unit (IMU) could be utilized to measure the accelerations of the robot chassis and therefore provide more accurate estimations of the movement velocities and pose of the robot.

7. CONCLUSION

This thesis studied an active sensing problem where three cameras can be operated in six configurations, and used as either monocular cameras or stereo pairs. A robot follows a trajectory defined as a sequence of visual landmarks at uncertain locations and has to direct its gaze to maximize information about the landmark locations. Parametric Gaussian representation is used for the robot's belief about landmark locations, which are assumed to be independent random variables. The problem was formulated as a POMDP with a reward function based on mutual information, and a myopic solution was provided. The algorithms were implemented on ROS using C++ and Python.

Simulation experiments demonstrate the benefits of using the proposed active sensing implementation over using fixed front-facing stereo pairs. The proposed active sensing solution outperforms the fixed camera solution when prior information is highly uncertain and the prior mean differs from the actual landmark positions. If prior information is accurate or the prior means happen to be close to the actual landmark positions, there is little difference between the two tested methods. There was, however, no disadvantage in using the active sensing in any of the tests performed.

The mutual information used as the reward function was approximated using Monte-Carlo methods, and simulation experiments were performed with multiple sample sizes to select a suitable range. Too few samples (< 50) result in poor optimization of gaze direction and erratic behavior, whereas with too many samples (> 500) the exploratory benefits of active sensing are somewhat lost.

The approach used in the thesis does have several limitations. The beliefs are represented with Gaussian distributions which might not be an acceptable approximation in some applications e.g. with multi-modal beliefs. The measurements from the cameras are considered to be two-dimensional, when in fact they are three-dimensional. In an environment where the landmarks or features that are extracted from the images are not at a fixed height, the use of three-dimensional measurements is vital. The landmarks are also considered to be independent random variables to simplify the problem. This is a clear shortcoming and prevents the use of loop-closures, which is an important method used in localization problems to minimize entropy and improve position estimates once a known landmark is encountered again. Moreover,

the dynamics model used was deemed to be unfit for real world application without further development.

Future work will aim to improve the shortcomings of the current approach, e.g. the solution will be generalized to Gaussian mixture models to handle multi-modal beliefs, and camera cones of observation will be defined in a three dimensional space. Applicability to real world situations can be improved by using an IMU to estimate robot movements. We also plan to apply similar active sensing methods to a wider range of problems in mobile robotics, such as multi-agent systems.

REFERENCES

Araya-López, Mauricio, Olivier Buffet, Vincent Thomas, and Françcois Charpillet (2010). "A POMDP Extension with Belief-dependent Rewards". In: *Advances in Neural Information Processing Systems*. Vol. 23. Curran Associates, Inc., pp. 64–72.

Bellman, Richard (1957). *Dynamic Programming*. Princeton University Press.

Bertsekas, Dimitri P. (1995). *Dynamic Programming and Optimal Control*. Vol. II. Athena Scientific.

Brooks, Alex M. (2007). "Parametric POMDPs for Planning in Continuous State Space". PhD thesis. The University of Sydney.

Cassandra, Anthony R. (1998). "Exact and Approximate Algorithms for Partially Observable Markov Decision Processes". PhD thesis. Brown University.

Chong, Edwin K. P., Christopher M. Kreucher, and Alfred O. Hero (2008). "POMDP Approximation Using Simulation and Heuristics". In: *Foundations and Applications of Sensor Management*. Ed. by Alfred O. Hero, David A. Castañón, Douglas Cochran, and Keith Kastella. Springer US, pp. 95–119.

Cover, Thomas M. and Joy A. Thomas (2006). *Elements of Information Theory*. 2nd ed. John Wiley & Sons, Inc.

Hartley, Richard and Andrew Zisserman (2004). *Multiple View Geometry in Computer Vision*. 2nd ed. Cambridge University Press.

Hauskrecht, Milos (2000). "Value-Function Approximations for Partially Observable Markov Decision Processes". In: *Journal of Artificial Intelligence Research* 13, pp. 33–94.

Hero, Alfred O. and Douglas Cochran (2011). "Sensor Management: Past, Present, and Future". In: *Sensors Journal, IEEE* 11.12, pp. 3064–3075.

Howard, Ronald A. (1960). *Dynamic Programming and Markov Processes*. MIT Press.

Ji, Shihao, Ronald Parr, and Lawrence Carin (2007). "Nonmyopic Multiaspect Sensing With Partially Observable Markov Decision Processes". In: *Signal Processing, IEEE Transactions on* 55.6, pp. 2720–2730.

Johnson, Leif, Brian Sullivan, Mary Hayhoe, and Dana Ballard (2014). "Predicting human visuomotor behaviour in a driving task". In: *Philosophical Transactions of the Royal Society of London B: Biological Sciences* 369.1636.

Kaelbling, Leslie Pack, Michael L. Littman, and Anthony R. Cassandra (1998). "Planning and acting in partially observable stochastic domains". In: *Artificial Intelligence* 101.1–2, pp. 99–134.

Kalman, Rudolf E. (1960). "A new approach to linear filtering and prediction problems". In: *Transactions of the ASME, Journal of Basic Engineering* 82.1, pp. 35–45.

Koenig, Sven (2001). "Agent-centered search". In: *AI Magazine* 22.4, p. 109.

Kreucher, Christopher M., Alfred O. Hero, and Keith Kastella (2005). "A Comparison of Task Driven and Information Driven Sensor Management for Target Tracking". In: *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC '05. 44th IEEE Conference on*, pp. 4004–4009.

Krishnamurthy, V. and D. V. Djonin (2007). "Structured Threshold Policies for Dynamic Sensor Scheduling—A Partially Observed Markov Decision Process Approach". In: *Signal Processing, IEEE Transactions on* 55.10, pp. 4938–4957.

Kullback, Solomon (1968). *Information Theory and Statistics*. Dover Publications Inc.

Kurniawati, Hanna, David Hsu, and Wee Sun Lee (2008). "SARSOP: Efficient Point-Based POMDP Planning by Approximating Optimally Reachable Belief Spaces". In: *Proceedings of Robotics: Science and Systems IV*.

Lauri, Mikko and Risto Ritala (2014). "Stochastic control for maximizing mutual information in active sensing". In: *ICRA 2014 Workshop: Robots in Homes and Industry: Where to Look First?*

Littman, Michael L. (1996). "Algorithms for Sequential Decision Making". PhD thesis. Brown University.

Martinez-Cantin, Ruben, Nando de Freitas, Eric Brochu, José Castellanos, and Arnaud Doucet (2009). "A Bayesian exploration-exploitation approach for optimal online sensing and planning with a visually guided mobile robot". In: *Autonomous Robots* 27.2, pp. 93–103.

Papadimitriou, Christos H. and John N. Tsitsiklis (1987). "The Complexity Of Markov Decision Processes". In: *Mathematics of Operations Research* 12.3, pp. 441–450.

Pineau, Joelle, Geoffrey Gordon, and Sebastian Thrun (2003). "Point-based value iteration: An anytime algorithm for POMDPs". In: *International joint conference on artificial intelligence*, pp. 1025–1032.

– (2006). "Anytime Point-Based Approximations for Large POMDPs". In: *Journal of Artificial Intelligence Research* 27, pp. 335–380.

Porta, Josep M., Nikos Vlassis, Matthijs Spaan, and Pascal Poupart (2006). "Point-Based Value Iteration for Continuous POMDPs". In: *The Journal of Machine Learning Research* 7, pp. 2329–2367.

Puterman, Martin L. (1994). *Markov Decision Processes—Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc.

Quigley, Morgan, Ken Conley, Brian P. Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y. Ng (2009). "ROS: an open-source Robot Operating System". In: *ICRA Workshop on Open Source Software*.

Raunio, Jukka-Pekka, Tuomas Välimäki, and Risto Ritala (2015). "Optimal Operation of a Three Camera System on a Four-wheel Robot". In: *XXI IMEKO World Congress "Measurement in Research and Industry"*.

Ross, Stéphane, Joelle Pineau, Sébastien Paquet, and Brahim Chaib-draa (2008). "Online Planning Algorithms for POMDPs". In: *Journal of Artificial Intelligence Research* 32, pp. 663–704.

Roy, Nicholas, Geoffrey Gordon, and Sebastian Thrun (2005). "Finding Approximate POMDP Solutions Through Belief Compression". In: *Journal of Artificial Intelligence Research* 23, pp. 1–40.

Särkkä, Simo (2013). *Bayesian filtering and smoothing*. Cambridge University Press.

Shani, Guy, Joelle Pineau, and Robert Kaplow (2013). "A survey of point-based POMDP solvers". In: *Autonomous Agents and Multi-Agent Systems* 27.1, pp. 1–51.

Silver, David and Joel Veness (2010). "Monte-Carlo Planning in Large POMDPs". In: *Advances in Neural Information Processing Systems*. Ed. by J.D. Lafferty, C.K.I. Williams, J. Shawe-Taylor, R.S. Zemel, and A. Culotta. Vol. 23. Curran Associates, Inc., pp. 2164–2172.

Smallwood, Richard D. and Edward J. Sondik (1973). "The Optimal Control of Partially Observable Markov Processes over a Finite Horizon". In: *Operations Research* 21.5, pp. 1071–1088.

Sondik, Edward J. (1978). "The Optimal Control of Partially Observable Markov Processes over the Infinite Horizon: Discounted Costs". In: *Operations Research* 26.2, pp. 282–304.

Spaan, Matthijs and Pedro Lima (2009). "A Decision-Theoretic Approach to Dynamic Sensor Selection in Camera Networks". In: *19th International Conference on Automated Planning and Scheduling (ICAPS-2009)*.

Spaan, Matthijs, Tiago Veiga, and Pedro Lima (2014). "Decision-theoretic planning under uncertainty with information rewards for active cooperative perception". In: *Autonomous Agents and Multi-Agent Systems*, pp. 1–29.

Spaan, Matthijs and Nikos Vlassis (2005). "Perseus: Randomized Point-based Value Iteration for POMDPs". In: *Journal of Artificial Intelligence Research* 24, pp. 195–220.

Stachniss, Cyrill, Giorgio Grisetti, and Wolfram Burgard (2005). "Information Gain-based Exploration Using Rao-Blackwellized Particle Filters". In: *Robotics: Science and Systems*.

Thrun, Sebastian (2000). "Monte Carlo POMDPs". In: *Advances in Neural Information Processing Systems 12*. Ed. by S.A. Solla, T.K. Leen, and K.-R. Müller. MIT Press, pp. 1064–1070.

Välimäki, Tuomas (2015). "Development of camera rig control in ROS-Linux environment". BSc thesis. Tampere University of Technology.

Van den Berg, Jur, Sachin Patil, and Ron Alterovitz (2012). "Efficient Approximate Value Iteration for Continuous Gaussian POMDPs." In: *26th AAAI conference on artificial intelligence*, pp. 22–26.

White, D. J. (1993). *Markov Decision Processes*. John Wiley & Sons, Ltd.