STOCHASTIC OPTIMAL CONTROL WITH APPLICATION IN VISUAL SERVOING

by

Aidin Foroughi

B.Sc., University of Tehran, 2010

A thesis presented to Ryerson University
in partial fulfillment of the
requirements for the degree of
Master of Applied Science
in the Program of
Mechanical and Industrial Engineering

Toronto, Ontario, Canada, 2013

Author's Declaration

I hereby declare that I am the sole author of this thesis or dissertation.

I authorize Ryerson University to lend this thesis or dissertation to other institutions or individuals for the purpose of scholarly research.

I further authorize Ryerson University to reproduce this thesis or dissertation by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

Abstract

STOCHASTIC OPTIMAL CONTROL WITH APPLICATION IN VISUAL SERVOING

Master of Applied Science, 2013

Aidin Foroughi

Department of Mechanical and Industrial Engineering Ryerson University

In this thesis, a new inference-based solution to stochastic optimal control (SOC) for general nonlinear systems is developed. This novel method applies to standard SOC problem, as well as robust and risk-seeking variations. The presented approach unifies many existing works, and makes possible, inference-based approximations to be applied to robust, risk-seeking, and standard SOC problems. Thus, an approximate method based on extended Kalman filtering is developed and tested on the inverted pendulum problem, and compared with existing methods. As an application, the developed algorithm was adapted to a practically important problem in visual control in robotics known as image-based visual servoing (IBVS). The proposed control methodology for visual servoing was implemented for real-time experiments, and was compared with the standard IBVS methodology. The experimental results show that the proposed method can improve the myopic behaviors of standard IBVS methodology.

Dedication

To my parents,
Farzad and Nahid,
and to my gaurdian angel,
Parna.

Contents

List of Tables

List of Figures

Nomenclature

Abbreviations

ADP	Approximate dynamics programming
BP	Belief propagation.
CoV	Calculus of variations
DP	Dynamic Programming
iLQG	Iterative linear-quadratic-Gaussian.
KL	Kullback–Leibler
LQ	Linear quadratic
MDP	Markov decision process
MIMO	Multiple-Input Multiple-Output
MP	Pontryagin's maximum principle
OC	Optimal control
OD	Optimal distribution
PP	Proximal point.
PPI	Posterior policy iteration.
PV	Probabilistic variational.
RDP	Robust dynamic programming.
RL	Reinforcement learning
ROC	Robust optimal control

SDDP	Stochastic differential dynamic programming.
SOC	Stochastic optimal control
VS	Visual servoing
KF	Kalman filter
LQ	Linear quadratic
LQR	Linear quadratic regulator
RBG	Rationality bounded game.

Mathematical Symbols

$\mathbf{L_s}$	Interaction matrix.
$\mathcal{A}$	Ambiguity set. The set of possible dynamics of the plant.
$c\left(\bar{\mathbf{x}}, \bar{\mathbf{u}}\right).$	Cost function, which is a function of state and control trajectories.
c_π	Cost incurred under policy π.
$, c_\nu$	The vertical coordinate of the center of the image, in image plane
c_ν	The horizontal coordinate of the center of the image, in image plane
$f\left(\mathbf{x}_{k+1} \mid \mathbf{x}_k, \mathbf{u}_k, k\right)$	The time-dependent probability of landing on state $\mathbf{x}_{k+1}$ from state $\mathbf{x}_k$ when the control signal is $\mathbf{u}_k$.
f_c	Focal length of the camera
$f_k\left(\mathbf{x}_{k+1} \mid \mathbf{x}_k, \mathbf{u}_k\right)$	The probability of landing on state $\mathbf{x}_{k+1}$ from state $\mathbf{x}_k$ when the control signal is $\mathbf{u}_k$.
${}^a\mathcal{F}_b$	The homogenous transformation of frame b w.r.t. frame a.
$J_i^\star\left(\mathbf{x}_i\right)$	The value function, which is the expected cost under the optimal policy.
$\mathbf{L_s}$	The interaction matrix
m	The dimension of the control space.
n	The dimension of the state space.
n_f	The number of features

${}^{a}\mathcal{P}$	The coordinates of a point w.r.t. coordinate frame a
$p^{*(\varepsilon,p_0)}$	The solution to a constrained optimal distribution problem, on a KL-ball with center p_0 and radius ε.
$p^{\star(\beta,p_0)}$	The Boltzmann disribution with degeneracy, p_0 and temperature β.
q_{μ,f,p_0}	The state-control trajectory distribution for dynamics f, policy μ and initial distribution,p_0.
$\mathbb{R}$	The space of real numbers.
${}^{a}\mathcal{R}_b$	The rotation matrix of frame b w.r.t. frame a
$\mathbf{s}$	Feature vector, which is composed of the 2D coordinates of a set of 2D features
$\mathbb{U}$	The control space, a subset the space of real-valued vectors with dimension m.
$\bar{\mathbf{u}}$	The control trajectory from the current time to the end of the control horizon, i.e., $\mathbf{u}_0, ..., \mathbf{u}_N$.
$\mathbf{u}$	Control signal. A real valued vector which specifies the control signal applied to the plant.
u	The horizontal coordinate of a 2D point on the image plane
v	The vertical coordinate of a 2D point on the image plane
$\mathbb{X}$	The state space, a subset the space of real-valued vectors with dimension n.
$\bar{\mathbf{x}}$	The state trajectory from the current time to the end of the control horizon, i.e., $\mathbf{x}_0, \mathbf{x}_1, ..., \mathbf{x}_N$
$\mathbf{x}$	State vector. A real valued vector which specifies the state of plant.
X	The x coordinate of a 3D point
Y	The y coordinate of a 3D point
Z	The z coordinate of a 3D point
$Z(\beta)$	The partition function.

Roman and Greek Symbols

$\delta(\cdot)$	The Dirac delta function.

ε_{f_k}	The radius of the KL-ball, defining the ambiguity set for the dynamics f_k.
ε_{f_k}	The radius of the ambiguity set of the dynamics.
$\mu^{\star o}$	The risk-seeking optimal policy.
$\mu^{\star r}$	The robust optimal policy.
$\mu^{\star r}$	The robust optimal policy.
$\mu_k\left(\mathbf{u}_k \mid \mathbf{x}_k\right)$	A stochastic policy, which is equal to $\Pr\left(\mathbf{U}_k = \mathbf{u}_k \mid \mathbf{X}_k = \mathbf{x}_k\right)$.
$\boldsymbol{\nu}$	Velocity screw, composed of stacking rotational and translational velocities
Ω	The rotational velociy vector
π	Control policy.
Π_D	The space of all deterministic policies.
Π_S	The space of all stochastic policies.
$\pi^\star$	The optimal policy.
π_k	A control policy at time step k. It's a function from state space to control space.

Miscellaneous

$\langle\cdot\rangle$	Expected value operator.
$\mathrm{KL}\left(p\left(x\right) \parallel q\left(x\right)\right)$	Kullback–Leibler divergence of $q(x)$ from $p(x)$.

Chapter 1

Introduction

1.1 Background and Origins

Stochastic optimal control (SOC) is a branch of modern control theory which deals with optimal control of systems with stochastic uncertainty. It is a powerful framework which allows one to consider various sources of uncertainty including measurement noise, stochastic process noise and parameter uncertainty, and can be viewed as the stochastic extension of *optimal control* (OC). Problems in OC date back to as early as 17th century [1] with the outset of developments in *calculus of variations* (CoV)[1]. By the 20th century, CoV had matured enough, through the works of Euler, Lagrange, Jacobi and Hamilton, to permit the development of a complete variational framework to OC. However, development of such a framework did not occur until the end of the second world war.

The post-war years constitute an important era in modern control, as very influential theories were introduced to deal with the emerging control and military applications. The classical frequency-domain and root-locus methods were gradually substituted by the more powerful state space (time-domain) models which simplified the representation of nonlinear, multiple-input multiple-output (MIMO) problems. Moreover, OC allowed direct implementation of performance criteria, such as fuel consumption and trajectory time, as cost functions to be minimized by the control.

Starting in 1950s, an important development on the Soviet side was a complete variational framework for OC introduced by Pontryagin, through what is known as the *Pontryagin's maximum principle* (MP), articulated in his seminal work [2]. Almost in parallel to Pontryagin, a second school emerged in the U.S. through the seminal works of Bellman [3, 4], who developed the theory of *dynamic programming* (DP) and the *Bellman principle of optimality*. Further developments of the same impact (if not more) were those by Kalman [5–7] who

[1] Calculus of variations is the mathematical field which deals with optimizating *functionals*. Functionals are mappings from functions to real values. Therefore, *a variational problem* (VP) is an optimization problem which picks a function as the optimal solution. For instance, finding the minimum energy trajectory of a robot can be expressed as a VP.

derived optimal control for *linear-quadratic* (LQ) problems[2] and identified the duality of filtering and control for *linear-quadratic* (LQ). His works resulted in what is today known as *LQ regulators* (LQR) and *Kalman filtering* (KF) which are of great theoretical and practical significance. Although both MP and DP are valid frameworks for deterministic OC problems, the former doesn't easily extend to stochastic settings while the latter does – and elegantly so. Therefore, since in this thesis our interest is only with optimal control of systems with random uncertainty, our focus will be on DP methodology.

The works of Bellman and Kalman mark the beginning of a large and ever-increasing body of works on (stochastic) OC which were contributed by researchers from a wide range of disciplines. Indeed, an optimal controller can be viewed as an intelligent decision maker which, given a cost function, a model of the system, and models of noise and uncertainty in the system, takes appropriate actions to minimize the cost. In this sense, (S)OC is a question of optimizing a *sequence of decisions*. Therefore, the research on (S)OC transcends control engineering and engages researchers in any field where decisions are important. Naturally, as a result of this fact, the literature on OC and SOC is rich in fields as diverse as artificial intelligence, operations research, management science, economics and finance. The applications are just as disparate and range from robotics to inventory management [8] to addressing U.S. financial debt crisis [9]. Although these problems have a common core, there are differences among them too; control theory and economics tend to work with continuous state/control spaces, whereas operational research and artificial intelligence deal with discrete, finite state/control spaces. The term, optimal control, is usually used when dealing with problems of the former type, while the latter problems are usually discussed under the title of *Markov decision processes* (MDPs). Nevertheless, in both discrete and continuous spaces, when the problems become larger, one faces serious challenges.

1.2 Challenges and Motivation

One of the most challenging issues in (S)OC is computational tractability (or intractability) of problems with high and/or continuous dimensions, which are typical of problems in robotics and, generally, in control engineering. In addition to problems in control engineering, large decision making problems can occur in a number of applications in energy, aerospace and resource allocation [10]. For these high-dimensional problems, approximate solutions to (S)OC play an indispensable role, as these problems couldn't be solved otherwise. Many researchers go as far as claiming that DP, while an important development, is useless without practical approximate solutions [10].

Indeed, from the time Bellman introduced the theory of DP up until today, developing efficient numerical solutions and tractable algorithms has always been a hot topic. The continuous growth of the literature on (S)OC can be partly attributed to the ever-increasing

[2] i.e. problems with linear dynamics and quadratic cost functions.

emergence of new applications for it, and partly associated with challenges of applying DP. The past few years, in particular, have seen increased interest in research on efficient and approximate methods, due to new breakthroughs in our perception of the relationships between SOC and *inference* [11–15]. Inference is a broad term, and different problems, such as finding posterior distributions, marginal distributions, or maximum a posteriori estimation and the filtering problem, all fall under the umbrella of inference problems. Inference also suffers from computational complexities which afflict SOC. Inference, as a field, has a very rich literature and many efficient approaches have been developed to deal with computational complexities of inference problems.

The new-found relations between inference and SOC has opened new doors for application of approximate inference methods to SOC problems [16–24]. More recently, the body of empirical evidence showcasing the success of inference-based solutions for real-world control and robotics applications has grown considerably [25–30].

Motivated by the importance of practical solutions to SOC, and following the success of recent inference-based solutions to SOC, this thesis is an attempt to develop a new approximate and efficient, inference-based solution to SOC.

1.3 Literature

Bellman identified and acknowledged the computational complexity of DP, referred to as the *curse of dimensionality*. Although toy problems with small dimensions may be easily solved using the current power of digital computers, for practical problems with large or continuous states, the exact solution based on DP is highly infeasible. Some problems, such as the ones in energy, have dimensions as large as tens or even hundreds of thousands [10]. Therefore, for practical and problems of interest, DP is "hard" to solve and the necessity for approximate methods is the rule rather than the exception.

The literature on approximate solutions to (S)OC and, more generally, to sequential decision making, is vast and convoluted. However, for our purposes, the literature can be broken down to four lines of work. The first line of work was that of Bellman and his community of operations researchers, who developed the theory of MDPs in their works [31–34]. Conventional MDP deals with sequential decision making problems when the spaces of states and actions are discrete and finite. An excellent review of the MDP theory can be in [35].

The second line of work was that of computer scientists who approached the problem of sequential decision making from an artificial intelligence perspective. Initial works were aimed at having computer programs *learn* to make optimal decisions, for instance, in games. This line of work is commonly referred to as *reinforcement learning* (RL). Although, RL initially started separately from other lines of work in approximate methods for decision making, later it turned out to have deep connections with some adaptive approximate DP

algorithms. Important references on RL include the excellent survey in [36] and the landmark textbook by Sutton and Barto [37]. Again, RL usually considers discrete and finite states and actions.

The third line of work is done by control theorists, dealing with continuous state/control space problems. Important works by this community include results by Werbos [38, 39], Bertsekas [40] and Tsitsiklis [41]. These works also established the connections among the literature on MDPs, RL and their own approximate control theoretic solutions. Important texts on the topic, from a control theoretic perspective, include the celebrated volume by Bertsekas and Tsitsiklis [42] and the subsequent textbooks by Bersekas [43]. In this thesis, in all of the above three lines of work are collectively referred to as *approximate dynamic programming* (ADP). Some other good textbook on ADP include [10, 44] and the second volume Bersekas two volume series [45], which he updates periodically and has made available for free online access.

The fourth and final line of works, which started only a few years ago, came mainly from computer scientists involved in the field of approximate inference methods. An important work is [12] which established a general duality between estimation and SOC. Subsequent works showed how SOC can be cast as an inference problem and solved using efficient inference algorithms [14, 16–22, 24]. Since inference is known to have deep relations with other fields such as statistical physics, thermodynamics, information theory, stochastic optimization and game theory [46–48], a consequence of relating SOC to inference is the implication that SOC is related to all these fields. A recent workshop, entitled *"The Statistical Physics of Inference and Control Theory"* [49], was aimed at discussing these links and connections. Such connections are not attractive merely from a theoretical point of view, but also from a practical one, as they enable us to use efficient algorithms developed separately in statistical physics and in inference to be applied to SOC.

This thesis is mainly motivated by the latter of these four lines of work for the following reasons:

- Computational efficiency: Several recent inference-based solutions show substantial improvements over previous works, in terms of computational efficiency. For instance, [17, 20, 28] report inference-based solution which is faster than older methods such as differential DP or policy gradient methods. Also, the work in [16] describes an algorithms which outperforms the most successful RL algorithms, in terms of computation time. Some other works in this area, sometimes referred to as *linearly solvable OC*, show how a subset of nonlinear SOC problems can be solved as linear problems, and therefore, solved much more efficiently than other ADP methods [13, 50, 51].

- Extensibility to other problems: In robotics and other control problems, aside from the control problem, one often faces several other problems such as state estimation, high-level reasoning and learning. These problems are often interwoven and interrelated, and can greatly benefit from interaction. Applying dissimilar and inconsistent frameworks

to each of these problems can make such interaction very hard or impossible. Instead, if one could integrate all this problems into a larger problem and treat them within a single framework, the resulting synergy could be substantial. Indeed, since high-level reasoning, learning and state estimation can be readily approached using inference, solving OC within an inference-based framework may bring us closer to an integration of all the above problems. Some examples of integrating high-level reasoning and optimal control of low-level robotic motions through an inference-based framework can be found in [30].

- Agreement with motor control theories: Motor control theories try to explain the mechanism governing the unparalleled motor abilities of biological systems. OC lays the foundations for some of the most notable theories in motor control. However, a recent paradigm shift in motor theories from OC to inference-based frameworks has received positive critiques from researchers in the field [52–55]. Accordingly, inference seems to be a better paradigm for explaining motor abilities of biological systems. Also, important problems in robotics, such as motion planning and control may be approached by mimicking and emulating biological mechanisms. Therefore, approximate inference-based solutions to SOC may be useful in realizing successful robotics applications.

Despite the above advantages, because of their recency, approximate inference-based solutions leave certain areas open for future work, including the following:

- Limited applicability: A large number of results in inference-based control can only be applied to a sub-class of SOC problems. For instance, in [13, 50, 51], only SOC problems with a special form of cost function[3] are addressed. Moreover, these works make certain assumptions regarding controllability of the system[4]. These requirements are hardly ever met by practical problems or even cases as simple as LQ problems [16].

- Analytical intractability: There exist few works in the literature which tried to overcome the limited applicability of the above inference-based solutions. For instance, Rawlik et al. [16] developed a general inference-based framework which applies to the largest class of SOC problems to-date. However, this method still lacks a tractable analytical form, because it doesn't lend itself to a simple closed-form solution even for LQ problems. A tacit evidence to support this deficiency is that the work presented in [16] lacks an implementation of their method for a benchmark control problem.

- Robustness issues: It is well-known that OC, in general, may be subject to robustness issues, because it fully trusts the available model of the system. Therefore, although

[3] i.e., SOC problems with a relative entropy term in the cost function.

[4] More specifically, they assume fully controllable dynamics, which is the assumption that the stochastic dynamics of the system may be brought into any desired form by control.

control maybe optimal for the available model, the optimality may be highly sensitive to the model of the system; a small misspecification of the model can result in highly sub-optimal solutions. There exists works that try to address robustness in OC, but they all result in computational complexity higher than the standard DP, and their additional dimensions exacerbate the curse of dimensionality. Currently, there is no inference-based robust optimal control methodology.

In short, most results in inference-based SOC, suffer from either limited applicability or analytical tractability. Motivated by the these issues, our work builds upon [16] and the related literature [12, 17, 22, 56], and offers a solution which enjoys applicability to general SOC problems and analytical tractability, along with several other advantages to be discussed. Our work is also inspired by some other works in the area such as, [57–59], which relate OC to thermodynamics and *rationality bounded game* (RBG) theory [47], and the robust control theory of Hansen-Sargent [60].

As a final note, it should be stressed that the four lines of work discussed above are by no means disjoint and there are big overlaps and parallels among them. As a result, our approach has fundamental connections to many other approximate solutions, therefore, the intent has been to faithfully reflect these connections. However, success in this matter is naturally limited as a result of the large amount of works in this area.

1.4 Contributions

In this thesis, a novel approximate solution to SOC is proposed and its applicability to problems in robotics is verified by applying it to visually guided robotics. Therefore, this thesis offers theoretical significance as well as practical value.

1.4.1 An Efficient Approximate SOC Solution

The significance of our solution comes from the fact that not only it addresses the aforementioned open problems in inference-based solutions to SOC, but it goes beyond that and provides additional generality and capabilities. Specifically, our solution offers the following:

- Generality: Unlike most other works in the area which have limited applicability, our solution can be applied to a large class of SOC problems. In fact, our solution constitutes the most general methodology in inference-based solutions, because among all inference-based methods, it can be applied to the largest class of SOC problems with continuous state/control spaces, arbitrary dynamics, and cost functions. Because of its generality, it can reproduce several methods as special cases [16, 56], including Rawlik's [16] recent work, which on its own is currently the most general inference-based solution in the literature.

- Analytical tractability: Unlike [16], our solution has a good analytical form which lends itself to closed-form solutions for a large class of problems in exponential families, such as LQ problems. This nice analytical form allows for LQ approximations to nonlinear-non quadratic.

- Robustness: While other inference-based solutions do not address robustness, our solution can produce *robust optimal control* (ROC) as a special case. Our ROC solution can handle unstructured uncertainties resulting in misspecification of the stochastic model.

- Flexibility: Our solution comes with an additional set of useful design parameters which enables the designer to fine-tune the behavior of the control to vary from robust, to optimal, to risk-seeking control. Moreover, these behaviors can be varied throughout the time, allowing the designer to define, for instance, a controller which is initially robust and conservative and gradually shifts towards optimality or risk-seeking control. To the best of our knowledge, such added flexibility is novel in control theory.

In addition to deriving a general solution with the above advantages, a closed-form solution is derived for the practical case of linearized dynamics and quadratized cost functions. This results in a new iterative solution similar to differential DP methods such as iLQG [61], except with much added flexibility, for instance with additional support for robustness and asynchronous updates. Finally, important connections to several related areas, such as RBG theory, robust control, inference, statistical physics and thermodynamics are established. These connections open up new doors for future research and improvements.

1.4.2 Visual Servoing via SOC

Visual servoing (VS) is control of robotic systems using visual feedback, i.e., through cameras and other visual sensors. VS is a well-studied problem which has been around at least as early as late 70s [62]. Since then, VS has become a standard technique in robot control with many industrial applications.

Our team in *robotics, mechatronics and manufacturing automation lab.* (RMAL) at Ryerson has a successful record in developing new theories and applications in VS [63–66] Our equipments and facilities enable us to apply and evaluate new control methodologies to problems in robotics including VS. SOC is especially well-suited for representing and solving VS, among other robotics problems. The reason is that VS is characterized by complicated MIMO nonlinear dynamics along with numerous sources of uncertainty, and SOC can naturally handle such nonlinear models in addition to *explicitly* accounting for different forms of uncertainty.

Conventional approaches to VS make extensive use of linearized models of the dynamics of the system along with other simplifying assumptions regarding uncertainties. In particular, stochastic uncertainties of the problem, such as noise in visual sensors, imperfect observability

of depth of the scene and other unmodeled random processes are usually ignored in *control level*. Moreover, as for the choice of control methodology, classical approaches to VS use very simple proportional type (P-type) controllers, which suffer from myopic behaviors. OC approaches do not suffer from such myopic behaviors, because they look further than a single time-step. Although there are a few instances of successful application of more advanced control methods to VS, including OC-based methods, SOC is the most powerful methodology to have yet to be applied to VS.

As another contribution of this thesis, VS is formulated as an SOC problem and our novel approximate method is utilized to derive a powerful solution for VS. In addition to the above, a simple one-step optimal controller, specifically for VS, with very low computational complexity. However, since it explicitly considers the uncertainty in the knowledge of the state of the system in control level, it will be shows, through both theory and practice, that it results in decreased noise sensitivity of VS, decreased sensitivity to imperfect knowledge of the depth and also increased accuracy of control.

VS is subject to many sources of uncertainty and our work addresses some of them. The uncertainties one faces in VS includes: noise in the images received from the cameras, changes in the illumination of the environment, noisy and imperfect control of the robot, imperfect knowledge of the calibration parameters of the camera and kinematics parameters of the robot, and imperfect knowledge of the depth of the scene. Also, there are always unmodeled effects, such as the the vibration of camera, the backlash of robotic joints and hysteresis, among other things.

Although all of the above can be potentially included in an SOC formulation, in this thesis only image noise, randomness in the control of the robot and uncertainty in the knowledge of the depth of the scene are considered. It will also be shows that our *robust optimal control* can provide robustness against unknown and unstructured errors the combined model of the camera and robot.

1.5 Notation and Other Conventions

A large part of this thesis is, to some extend, mathematically inclined and it helps to provide a brief discussion on the mathematical notation and other conventions used herein. Italic symbols $(x, u, f(\cdot))$ are used to represent scalar variables and functions. Bold symbols $(\mathbf{x}, \mathbf{u}, \mathbf{f}(\cdot))$ represent vectors, matrices, a vector-valued functions.

For representing probability and conditional distributions, a common notation found in many probability theory textbooks is used in this thesis. Specifically, $p(x)$, represents the probability of an underlying random variable, X, having the value, x. Or in mathematical terms, $p(x) = \Pr(X = x)$. Similarly, $p(y) = \Pr(Y = y)$. Although the above may seem natural and commonplace, there is a peculiarity to it. To see this fact, it should be noted that normally a function is defined by its symbol and the number of its arguments. That is,

if $f(x) = x^2$, then $f(y)$ equals y^2, which is why sometimes a function is represented without explicitly defining it's arguments, such as, $f(\cdot) = (\cdot)^2$.

The peculiarity of the p notation in probability theory comes from the fact that the function, $p(\cdot)$, changes according to its argument, e.g., $p(y)$, is not equal to $p(x)|_{x=y}$, but rather, it is a different function, defined as $p(y) = \Pr(Y = y)$. In other works, with this commonly used loose notation in probability theory, the function p is a special function which is defined, not only by the number of arguments, but also by the variables passed to it as arguments. With this clarification, despite the fact the p notation enjoys less mathematical rigor, since it greatly enhances readability and because of its popularity, it is adopted in this thesis.

1.6 Structure of the Thesis

- Chap. 2 reviews the theory of DP and important concepts therein and explains computational complexity of DP, known as the curse of dimensionality.
- Chap. 3 Motivated by these computational challenges, details our new approximate solution to SOC and describes its relations and improvements to existing works.
- Chap. 4 explains the application of our SOC solution to VS and derives special controllers, discusses improvements over basic VS methods and discusses experimental results.
- Chap. 5 concludes the thesis with a discussion on the results and potential directions for future research.

Chapter 2

The Classics: Dynamic Programming

2.1 Introduction

> *"The 1950s were not good years for mathematical research ...Hence, I felt I had to do something to shield the Air Force from the fact that I was really doing mathematics ... What title, what name, could I choose? I was interested in planning, in decision making, in thinking. ... I decided therefore to use the word, 'programming.' I wanted to get across the idea that this was dynamic, this was multistage, this was time-varying—I thought, let's kill two birds with one stone. Let's take a word that has an absolutely precise meaning, namely dynamic, in the classical physical sense ... and it's impossible to use the word, dynamic, in a pejorative sense. Try thinking of some combination that will possibly give it a pejorative meaning. It's impossible. It was something not even a Congressman could object to. So I used it as an umbrella for my activities."*
>
> From the autobiography of Richard E. Bellman: Eye of the Hurricane.

This chapter is an attempt to present a brief overview of the most important concepts and ideas in SOC and the Bellman theory of DP, hopefully, without resorting to a duplication of classical theorems and results in the field. Our emphasis is to review the basic concepts and definitions in DP and to show the many sources of difficulty in obtaining the optimal solution. Since contributions to OC come from researchers from various fields, the terminologies and notations used in the literature are different. For instance, one can encounter *control signal, action* and *decision* in different contexts, while in fact, they all refer to the same concept. For consistency, the terminology and the language of control theory is used throughout this chapter, with a few exceptions when some terms are borrowed from other contexts.

The chapter starts with a problem definition of SOC with *full state observability* and continuous state/control spaces. The Bellman principle is discussed and DP solution is derived. The computational challenge of applying DP, also known as the *curse of dimensionality,* is discussed in some depth. This chapter is concluded with a high-level discussion on the ideas behind some important approximate solutions to SOC, and how they compare to our approach.

2.2 Stochastic Optimal Control with Full Observability

Most textbooks on the subject of DP start their discussion with deterministic formulations, then extend it to stochastic settings. However, since our interests lie solely with stochastic problems, we delve right into the subject matter. For a review of deterministic OC see [43]. What follows is the basic definitions used throughout this chapter and the rest of the thesis. These definitions are consistent with the definition used in the literature concerning MDPs with continuous state and control spaces.

2.3 Definitions

Let us start by reviewing the basic definitions and terminology in OC.

Plant: The system to be controlled. It is assumed that a (nominal) mathematical model of the *plant* is available. This mathematical model describes the evolution of the *state* of the plant.

State vector: The state of the plant is described by a finite dimension real-valued vector, $\mathbf{x}$. *State vector* encodes all the information about the system which is of importance to the control engineer. The space of all possible states of the system, or the *state space*, is denoted by $\mathbb{X}$. Therefore, $\mathbf{x} \in \mathbb{X} \subseteq \mathbb{R}^n$, where n is the the dimension of the state space.

Control signal: The plant is controlled using a *control signal*, $\mathbf{u}$, which is a finite dimension real-valued vector. The space of all possible control signals is referred to as the *control space*, denoted by $\mathbb{U}$. Again we have,$\mathbf{u} \in \mathbb{U} \subseteq \mathbb{R}^m$, where m, is the dimension of the the control space.

Dynamics: The *dynamics* of the plant mathematically describes the evolution of the state of the plant given the control signal. In this thesis, the focus is mainly on *state space*

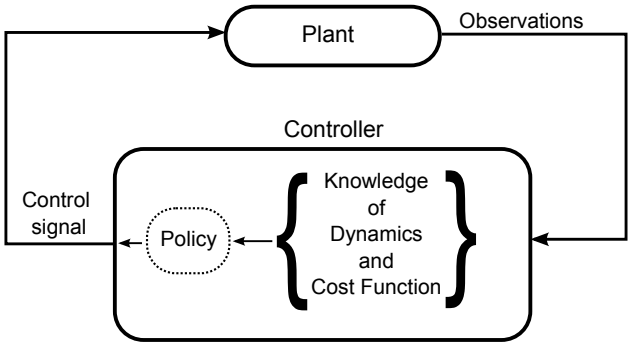


Figure 2.1 – A general template depicting different types of optimal control problems and the relationship among the basic concepts and terms in our problems.

models in discrete-time and continuous state and control spaces. The reason is that these models appear frequently in most control problems and also because, ultimately, they are more amenable to a digital implementation. One of the ways to describe *stochastic* dynamics of a plant is by using *transition function*s. A transition function is the (conditional) probability that the plant transitions to state $\mathbf{x}_{k+1}$ from $\mathbf{x}_k$ as a result of applying the control signal $\mathbf{u}_k$. In mathematical terms, dynamics is defined by the following time-dependent transition function,

$$f\left(\mathbf{x}_{k+1} \mid \mathbf{x}_k, \mathbf{u}_k, k\right) = f_k\left(\mathbf{x}_{k+1} \mid \mathbf{x}_k, \mathbf{u}_k\right) \triangleq \Pr\left(\mathbf{X}_{k+1} = \mathbf{x}_{k+1} \mid \mathbf{X}_k = \mathbf{x}_k, \mathbf{U}_k = \mathbf{u}_k\right), \tag{2.1}$$

where k denotes the discrete time index. It should be noted that the definition of dynamics through transition functions implicitly encodes all sources of randomness in the processes governing the dynamics of the plant.

Observations: It is assumed that some property of the plant is constantly *observed*. In general, these observations may be the *exact* state of the plant, exact measurements of a function of the state, or some noisy variant of these two, depending on the type of the problem at hand. However, in this chapter and Chap. 3, it is assumed that these observations constitute exact measurements of the state of the system. That is, at time step k, the complete information about the state, i.e.,$\mathbf{x}_k$, is available. This assumption is referred to as the *full observability* assumption.

Cost function: The desired behavior of the plant is described through the cost function. Cost functions may arise naturally in some problems, while they may also be *designed* by the control engineer, such that they encode the desired behavior of the plant. For instance, if the desired behavior of control is to minimize the energy consumption of the plant, one can define the cost function to be the energy consumption of the plant, or an approximation of it. In general, the designer can define a cost function with higher values around undesirable states and lower values around desired regions of the state space. Hence, minimizing the cost function is meant to drive the plant to desired regions of the state space, and thus, result in the desired behavior of the plant. The cost function, in mathematical terms, is a real-valued function of the current and future values of state and control signal. That is, the cost function is of the form,

$$c(\mathbf{x}_0, \mathbf{x}_1, ..., \mathbf{x}_N, \mathbf{u}_0, \mathbf{u}_1, ..., \mathbf{u}_N), \tag{2.2}$$

c describes the control cost from time step 0 to time step N, and N is referred to as the *control horizon*. When N is infinite, one has an infinite horizon OC problem. For simplicity of notation, the state and control trajectories are summarized into $\bar{\mathbf{x}}$ and $\bar{\mathbf{u}}$

. Therefore, the control cost can be defined as,

$$c\left(\bar{\mathbf{x}}, \bar{\mathbf{u}}\right). \tag{2.3}$$

An important class of cost functions are those which *decompose* into separate terms for each time step, i.e.,

$$c\left(\bar{\mathbf{x}}, \bar{\mathbf{u}}\right) = \sum_{k=0}^{N} c_k\left(\mathbf{x}_k, \mathbf{u}_k\right). \tag{2.4}$$

Most monographs on the subject [43, 67] simply start with decomposable cost functions, while it should be noted that for general cost functions, DP does not apply. The reason is that in general case, the problem cannot be broken into substructures due to coupling between decisions. Indeed, the decomposition of the cost function gives the problem a *Markov* property which admits a separation of the cost to an instantaneous cost and a future cost. Without this separation, the DP solution is not possible to derive. This is why for the rest of this thesis, only decomposable cost function are considered.

Control policy: The controller needs to decide which control signal to choose in order to minimize the cost, based on all information available to it. Mathematically, *policy*,π , is a time-dependent mapping from the state space of the plant to the control space,$\pi : \mathbb{X} \times t \mapsto \mathbb{U}$. In other words, policy[1] defines control signal as a function of time and the state of the plant,

$$\mathbf{u}_k = \pi\left(\mathbf{x}_k, k\right) \triangleq \pi_k\left(\mathbf{x}_k\right), \tag{2.5}$$

where,π_k, which is the policy at time step, k, is referred to as a *decision*. In this sense, for discrete-time problems, policy consists of a sequence of decisions. It should be immediately clear that control policy is, indeed, very closely related to the concept of *control law* in feedback control theory.

The above definitions and the relations among them are depicted in Fig. 2.1 to visually describe a general SOC problem. Having the above definitions, the next step is to review Bellman principle of optimality and DP.

2.4 Bellman's Principle of Optimality and Dynamic Programming

The optimal controller is one that implements the optimal policy. Therefore, optimality should be defined more accurately. It was previously stated that the objective of OC is to minimize the cost function. However, because of the stochasticity of the dynamics of the

[1]What is presented in this chapter is more accurately called a *deterministic* policy, because it specifies a *single* control signal for each observed state. In the next chapter, the concept of stochastic policy will be presented, where for each observed state, a random distribution of control signals is given by the policy.

system, one cannot predict the incurred cost of a policy. In other words, the cost incurred under policy, π,

$$c_\pi = \sum_{k=0}^{N} c_k\left(\mathbf{x}_k, \pi_k\left(\mathbf{x}_k\right)\right), \tag{2.6}$$

is a random variable. Therefore, one cannot directly minimize the incurred cost. In SOC, one chooses to minimize the *expected value* of the incurred cost[2]. Therefore, the optimal policy is defined as the policy, under which, the *expected cost* is minimized. In mathematical terms,

$$\pi^\star = \arg\min_\pi \left\langle \sum_{k=0}^{N} c_k\left(\mathbf{x}_k, \pi_k\left(\mathbf{x}_k\right)\right) \right\rangle, \tag{2.7}$$

where, $\langle\cdot\rangle$, is the expected value operator, and $\pi^\star$ is the *optimal policy*. Because of the decomposability of cost function and linearity of expected value operator, the above problem can be broken into smaller sub-problems. *Bellman's principle of optimality,* introduced in his seminal works [3, 4], describes the relationship between these sub-problems and the original problem.

2.4.1 Bellman's Principle

Theorem 1. *"An optimal policy has the property that, whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision."*. [3]

Proof. See [43]. □

In mathematical terms, the principle of optimality states that if $\pi^\star = (\pi_0^\star, \pi_1^\star, ..., \pi_N^\star)$ constitutes the optimal policy for a problem starting at time 0 to N, the truncated policy $\left(\pi_i^\star, \pi_{i+1}^\star, ..., \pi_N^\star\right)$ constitutes the optimal policy for the truncated problem starting at time i, until N. As a consequence of Bellman's principle of optimality, one can break the original problem of finding $\pi^\star$ into smaller truncated problems, knowing that the optimal policy for these truncated problems make up the optimal policy for the original problem. In fact, DP is nothing but the application of this principle. However, before going about describing DP completely, let us define the concept of *value function* (or *optimal cost-to-go*).

Value function: Value function is defined as the expected cost incurred under the optimal policy, i.e.,

[2]The expected value of the cost is not the only plausible objective in the stochastic setting. Other possibilities include the risk-sensitive objective [68], which tries to minimize a linear combination of the expected cost and the variance of the cost. This objective is useful in problems where the variance of the cost are important as well, i.e. for cost sensitive problems. With the same logic, higher cumulants of the cost variable may enter the new definition of the cost function [69]. Ultimately, one can imagine a *cost density shaping* framework, with the objective of completely shaping the distribution of the cost [70].

$$J_i^\star(\mathbf{x}_i) = \left\langle \sum_{k=i}^{N} c_k\left(\mathbf{x}_k, \pi_k^\star(\mathbf{x}_k)\right) \right\rangle. \tag{2.8}$$

Value function is important because one can derive the optimal policy from value function by,

$$\pi_i^\star(\mathbf{x}_i) = \arg\min_{\mathbf{u}_i} \left\{ c_i(\mathbf{x}_i, \mathbf{u}_i) + \left\langle J_{i+1}^\star(\mathbf{x}_{i+1}) \right\rangle \right\}, \tag{2.9}$$

where the expected value is given by,

$$\left\langle J_{i+1}^\star(\mathbf{x}_{i+1}) \right\rangle = \int_{\mathbf{x}_{i+1}} f_i(\mathbf{x}_{i+1} \mid \mathbf{x}_i, \mathbf{u}_i)\, J_{i+1}^\star(\mathbf{x}_{i+1})\, d\mathbf{x}_{i+1}. \tag{2.10}$$

Therefore, finding the value function leads one to the optimal policy. In fact, DP finds the value function as an intermediate step, instead of directly seeking the optimal policy. The reason for this intermediate step is that the minimization in Eq. 2.7 is over decisions, which are *functions*. This means that the minimization problem in Eq. 2.7 is a variational problem. However, the minimizations described by Eq. 2.9 are over control signals, which are vectors. Therefore finding the value function as an intermediate step turns the original variational problem into a regular optimization problem which, in general, can be solved much easier. We will shortly see that DP finds the value function for truncated problems and applies Bellman's principle to find the optimal policy for the original problem.

2.4.2 Dynamic Programming

Bellman's principle suggests that one can start off with smaller sub-problems and reuse the solutions to make up the solution to the original problem. The smallest sub-problem for a finite horizon problem is the one with only the last time step. It's easy to see that the value function at the last time step is,

$$J_N^\star(\mathbf{x}_N) = \min_{\mathbf{u}_N} \{ c_N(\mathbf{x}_N, \mathbf{u}_N) + 0 \}. \tag{2.11}$$

After finding the value function for the smallest sub-problem, one can move back one-step and find the value function at time $N-1$, i.e.,

$$J_{N-1}^\star(\mathbf{x}_{N-1}) = \min_{\mathbf{u}_{N-1}} \left\{ c_{N-1}(\mathbf{x}_{N-1}, \mathbf{u}_{N-1}) + \langle J_N^\star(\mathbf{x}_N) \rangle \right\}, \tag{2.12}$$

where,

$$\langle J_N^\star(\mathbf{x}_N) \rangle = \int_{\mathbf{x}_N} f_i(\mathbf{x}_N \mid \mathbf{x}_{N-1}, \mathbf{u}_{N-1})\, J_N^\star(\mathbf{x}_N)\, d\mathbf{x}_N. \tag{2.13}$$

At this point, it's clear that one can continue solving for the value function at time step $N-2$ and so on. Therefore, the following backward recursion solves the value function for

all steps, starting at the time horizon and sweeping backwards.

$$J_i^\star(\mathbf{x}_i) \quad = \quad \min_{\mathbf{u}_i} \left\{ c_i(\mathbf{x}_i, \mathbf{u}_i) + \left\langle J_{i+1}^\star(\mathbf{x}_{k+1}) \right\rangle \right\}. \tag{2.14}$$

This backward recursion is nothing but the DP solution which was to be derived. The optimal policy is then given by,

$$\pi_i^\star(\mathbf{x}_i) = \arg\min_{\mathbf{u}_i} \left\{ c_i(\mathbf{x}_i, \mathbf{u}_i) + \left\langle J_{i+1}^\star(\mathbf{x}_{i+1}) \right\rangle \right\}. \tag{2.15}$$

2.4.3 Complexity of Dynamic Programming

The DP algorithm discussed in the previous requires a minimization of cost function for each time step, over the space of control inputs, $\mathbb{U}$, for $\forall \mathbf{x}_i \in \mathbb{X}$. This minimization can result in several difficulties as follows:

- Curse of modeling: The dynamics of the plant may have a highly nonlinear and complex form, which suggest that the DP may not be carried out in closed form and numerical minimization may be needed. The trade-off between the accuracy of modeling and the increased computational effort is sometimes referred to as the *curse of modeling*.
- Curse of dimensionality: A numerical solution of each minimization in DP is prone to curse of dimensionality, because it has an exponential complexity with respect the dimensions of the state and the control signal.

To further explain the curse of dimensionality, one can observe that in each iteration of the Bellman recursion, an expected value operation and a minimization operation needs to take place. In general case, these nested expectation and minimizations are largely considered to be intractable [71]. The expectation is an integration on the state space, $\mathbb{X}$, and the minimization is over the control space,$\mathbb{U}$. Both of these spaces grow exponentially w.r.t. their dimensions. It is important to note that for special cases, for instance, unconstrained linear quadratic (LQ) problems, the minimization and expectation can be performed analytically, and therefore the curse of dimensionality doesn't exist [6].

2.4.4 Robustness

As mentioned in Chap. 1, SOC may suffer from robustness issues, because by assumption it fully trusts the nominal model of the dynamics of the plant. The optimal policy minimizes the expected cost for the nominal dynamics, but it does not say anything about the true dynamics of plant, which might be slightly different from the available model. Therefore, perturbations in parameters of the dynamics may render the optimal policy severely sub-optimal or even infeasible [72]. As a result, all exact and approximate solutions to standard

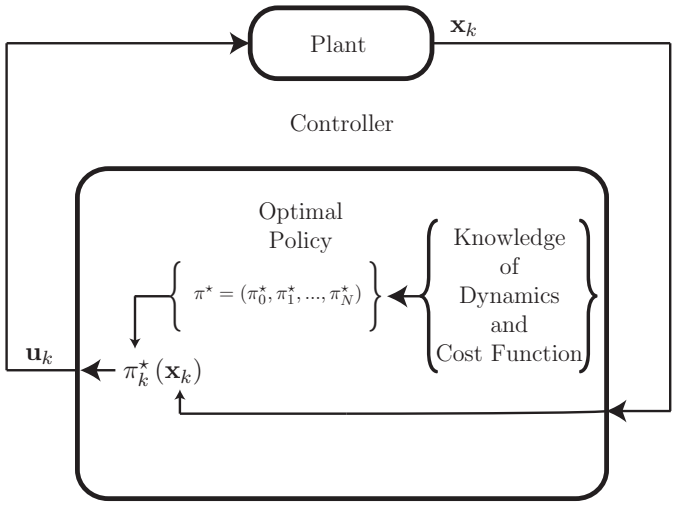


Figure 2.2 – A diagram depicting SOC problems with full observability.

SOC suffer from the same robustness problem, because their underlying assumptions are the same.

In the next section, a high-level overview of some of the remedies for computational complexity and robustness problems are discussed.

2.5 Related Works and Our Approach

Previously, the curse of dimensionality was discussed and it was shown how it can limit the use of DP for many practical problems. To tackle the curse of dimensionality, a rather obvious idea is that instead of taking the expectation and minimization over the full spaces of state and control, one can reduce these operations to a small relevant region or subset of them [45]. Methods based on function approximations comprise a large part of the works in ADP. In these methods, either the value function or the policy are approximated by different function approximation methods such as neural-networks or linear functions, resulting in approximate value iteration [42] or approximate policy iteration [73] methods. However, almost all of these works are best applicable to discrete and finite state/control spaces, and are not easily applicable to continuous spaces.

Among approaches which are applicable to continuous space SOC problems, one can name *stochastic differential dynamic programming* (SDDP) methods [74], such as the very successful iterative linear-quadratic-Gaussian (iLQG) approach [61]. iLQG is based on the idea of solving the DP for a small region of the state and control spaces, by assuming an initial state-control trajectory and solving the DP around this initial guess. Furthermore, iLQG uses linearized dynamics and quadratic approximations to the cost function around the initial trajectory, and therefore, it can solve DP analytically. Therefore, it achieves second order convergence rate to local optimal solutions. Because of its empirical success

for problems with large and continuous state and control spaces, iLQG has also been used as ground truth and for comparison in several works and publications [16, 28] Moreover, because of its computational efficiency and fast convergence, it's a useful benchmark for comparing the computational load of new algorithms.

To address robustness, several works have developed what is known as *robust dynamic programming* (RDP) [75, 76], which is a robust counterpart of the DP algorithm seen above. However, RDP is even more computationally demanding than the standard DP. This fact limits its applicability to problems which can be quickly solved with standard DP. However, for our applications, DP is too hard, and consequently, RDP is also not applicable.

To address computational efficiency and robust control, our work formulates a more general version of SOC (which includes robust control) as an inference problem. Casting SOC as an inference problem does not immediately solve the curse of dimensionality but it opens up possibilities of applying efficient approximate inference problems to SOC. This idea has already been explored for standard SOC with much empirical success for robotics applications[16, 28]. However, better analytical solutions are possible and, more importantly, robust control yet remains to be addressed.

After formulating the problem as an inference problem, approximations used in extended Kalman smoothing are borrowed from the inference field and applied to SOC. Because extended Kalman smoothing also assumes linearized dynamics and quadratic log-likelihood functions, our solution turns out to be very similar to an improved version of iLQG. In fact, iLQG can be assumed as a special case of our approach. Furthermore, extended Kalman smoothing approximation is just one of the many approximations used in inference. Namely, the sort of approximations used in unscented Kalman filters or particle filters could potentially be used to solve the SOC, formulated as an inference problem.

Chapter 3

Inference for Optimal Control

3.1 Introduction

In Chaps. 1 and 2 several challenges in SOC were pointed out. Namely, computational complexity and the curse of dimensionality were discussed in the context of standard DP and the robustness issues associated with the standard SOC were discussed in some depth. In this chapter, the goal is to derive a solution to SOC which enjoys flexibility, generality, computational efficiency and robustness.

It will be shown that ultimately a probabilistic variational (PV) formulation of SOC problems lends itself to an iterative and closed-form solution which enjoys all the above characteristics. Variational calculus serves as the hub connecting many approximate and exact methods for solving various problems in different contexts, such as statistical physics, game theory, information theory and inference. Therefore, adopting a variational approach enables us to better appreciate the connections of SOC to problems within different contexts and the existing approximate solutions therein. History has shown that revealing the connections between two fields is often mutually beneficial and can lead to major breakthroughs. For this reason, in this chapter the relationships between our approach to SOC and other fields will be made clear. Specifically, it will be shown that our method is related to some approximate inference methods such as *belief propagation* (BP) [77], and has close relationships with *rationality bounded games* (RBGs) of Wolpert [47], *robust control* framework of Hansen and Sargent [60], and inference-based solutions to control theory [16, 23, 27, 28]. These relationships may prove useful for future developments.

3.1.1 Roadmap and Contributions

In Sec. 3.2, the concept of *stochastic policy* is introduced by extending the concept of deterministic policy seen in Chap. 2, similar to [16]. The reason for introducing stochastic policies is that the problem of finding *stochastic* optimal policies may be solved in a *continuous* and unconstrained domain, and it will be shown that under certain assumptions, stochastic

policies may be found in closed-form. This wouldn't be possible, had we constrained ourselves to deterministic policies.

In order to address robustness and also to increase generality and flexibility of our solution, in Sec. 3.3 three problem formulations including standard, robust and risk-seeking SOC are defined, in terms of variational problems seeking *stochastic* optimal policies. These problems have been considered by other works in the literature and addressed separately. In this chapter, although a common formulation with previous works is used, a novel solution to the problem is derived. Specifically, It will be shown that a *single* solution based on the proximal point approach [78, 79] for all three problems may be derived, and that the designer may smoothly switch the behavior of the controller from standard to robust, to risk-seeking SOC.

Before deriving the proximal point solution, however, it is necessary to review some preliminary variational problems and see how they can be approached. This is done Sec. 3.4. These basic problems can be found throughout the literature as basic lemmas [16, 51, 58], but they are gathered here for completeness and for a quick review. Secs. 3.5 and 3.6 constitute most of the contribution of this chapter, where the basic lemmas introduced earlier are applied to the three variations of SOC problems, and a general closed-form iterative solution is derived.

Next, some related works in the field are reviewed in Sec. 3.8 and their relationship to our work is studied in some detail. It will be shown that many of these works can be considered as special cases or variations of our approach. In Sec. 3.7 our general solution is approximated by making assumption similar to those made in the context of extended Kalman filtering, to give a practical solution for nonlinear problems. Finally, in Sec. 3.9 it is shown how inference for optimal control and inference for state estimation can be combined to yield an integrated estimation-control framework.

The core contribution of this chapter is:

- A proximal point formulation of several variations of SOC and the subsequent unified closed-form solution.

Other contributions include:

- Deriving a unified Gaussian message-passing solution for stochastic optimal/robust/risk-seeking control.
- Presenting an integrated estimation-control method.
- Unifying numerous works in the area of inference-based control.

Aside from the novelties above, a unified overview of several basic but important variational problems along with their solutions and relations is presented in this chapter. This overview is valuable because although these problems and their solutions constitute the cornerstone of many works in the field, a useful and self-sufficient overview of them doesn't exist.

3.2 Stochastic Policies

The policies encountered in Chap. 2 (and also in conventional literature concerning MDPs and SOC) can be thought of as special cases of a more general class of policies. Previously, policy was defined using functions which mapped each observed state to a control signal, i.e.,

$$\mathbf{u}_k = \pi_k(\mathbf{x}_k). \tag{3.1}$$

Given an observed state, $\mathbf{x}_k$, the (deterministic) policy function, $\pi_k(\mathbf{x}_k)$, specifies a *single* control signal to be applied to the plant. However, here, the definition of policy is extended such that the policy can propose a *distribution* over the control signals, for each observed $\mathbf{x}_k$. Such a policy is known as a *stochastic* (or randomized) *policy*. In mathematical terms, a stochastic policy is defined by a (time-dependent) conditional probability function,

$$\mu(\mathbf{u}_k \mid \mathbf{x}_k, k) \triangleq \mu_k(\mathbf{u}_k \mid \mathbf{x}_k) \triangleq \Pr(\mathbf{U}_k = \mathbf{u}_k \mid \mathbf{X}_k = \mathbf{x}_k). \tag{3.2}$$

With this definition, deterministic policies become the special case when these distribution collapse to delta Dirac functions, i.e., when they become concentrated on a single control signal. More specifically, a deterministic policy, μ^D, is the special case when,

$$\mu_k^D(\mathbf{u}_k \mid \mathbf{x}_k) \quad = \quad \delta(\mathbf{u}_k - \pi_k(\mathbf{x}_k)), \tag{3.3}$$

where $\delta(\cdot)$ is the Dirac delta function. The set of all possible deterministic policies is denoted by Π_D, and the set of all stochastic policies by Π_S. Clearly, $\Pi_D \subset \Pi_S$. It is well-understood that in a decision making problem with a deterministic cost function, no stochastic policy is superior to the optimal deterministic policy [80, 81]. Therefore, if the goal is to find the *exact* optimal policy, one could limit the search for the optimal policy to Π_D. This is why it is common in MDP to narrow down the search to the class of deterministic policies [43].

However, when the ultimate goal is to find good *sub-optimal* policies, there is no reason to rule out stochastic policies, because a sub-optimal stochastic policy may still do better than a sub-optimal deterministic policy. Fig. 3.1 compares a sub-optimal stochastic policy and a sub-optimal deterministic policy against the optimal determinist policy and illustrates how a stochastic policy may be *closer* to the optimal policy than a deterministic one. Furthermore, since $\Pi_D \subset \Pi_S$, increasing the scope of problem from Π_D to Π_S causes no loss of generality. In fact, if a policy is found to be superior to all other policies in Π_S, it is also superior to all policies in Π_D, and therefore, it is bound to be either the deterministic optimal policy or an equally good stochastic policy. This fact has been recognized in [16], while stochastic policies have also been used in the literature for other reasons [81].

The last remaining question is that, given a good stochastic policy, how one can *execute*

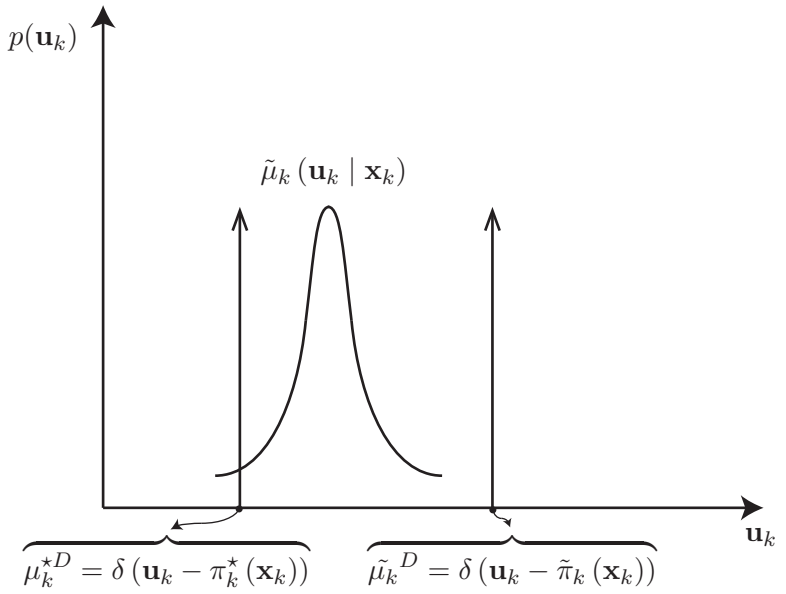


Figure 3.1 – Comparison of sub-optimal stochastic and deterministic policies against the optimal deterministic policy. $\tilde{\mu_k}^D$ and $\tilde{\mu_k}$ are sub-optimal deterministic and stochastic policies, respectively, and $\mu_k^{\star D}$ is the optimal policies.

it. Instead of proposing a single control signal to be applied to the plant, a stochastic policy proposes a distribution over control signals, and a distribution can't be *directly* applied to the plant. Therefore, an extra sampling step is needed before one can execute a stochastic policy. As for the sampling, one can either generate a random sample from the distribution $\mu_k(\mathbf{u}_k \mid \mathbf{x}_k)$ or deterministically choose this sample to be, for instance, the mean, the mode, or the median of $\mu_k(\mathbf{u}_k \mid \mathbf{x}_k)$.

With this discussion, from this point on, the goal is to find an efficient solution for sub-optimal stochastic policies. It will be shown that under certain conditions, sub-optimal stochastic policies can be derived analytically and in closed-form. But before that, it is useful to take a look at the expected value of the cost under a stochastic policy. It's easy to see that the expected cost is given by,

$$\langle c(\bar{\mathbf{x}}, \bar{\mathbf{u}}) \rangle = \int_{\bar{\mathbf{x}},\bar{\mathbf{u}}} p_0(\mathbf{x}_0) \prod_{k=0}^{N} \mu_k(\mathbf{u}_k \mid \mathbf{x}_k) f_k(\mathbf{x}_{k+1} \mid \mathbf{x}_k, \mathbf{u}_k) c(\bar{\mathbf{x}}, \bar{\mathbf{u}}) d\bar{\mathbf{x}} d\bar{\mathbf{u}}, \tag{3.4}$$

where, $p_0(\mathbf{x}_0)$ is the distribution of the initial state, i.e.,

$$p_0(\mathbf{x}_0) = \Pr(\mathbf{X}_0 = \mathbf{x}_0). \tag{3.5}$$

One may notice that the expected cost is, in fact, a weighted integral of the cost function, where the weighted by the probability distribution of state and control trajectories. This probability distribution is generated by applying the policy, μ, to the dynamics, f, starting

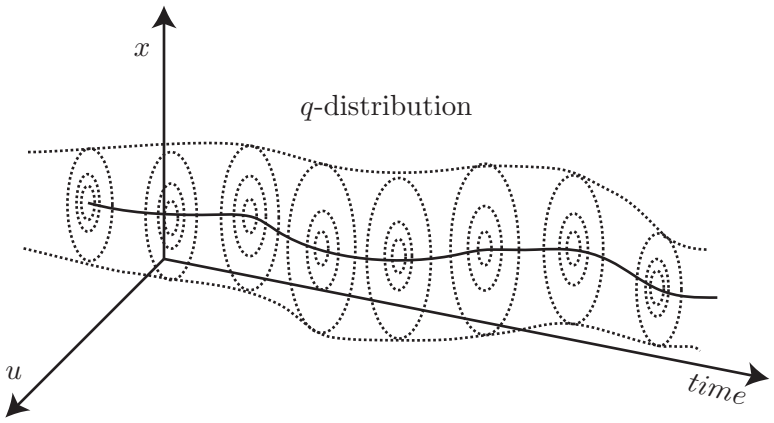


Figure 3.2 – An illustration of a q-distribution for a one dimensional state and control system.

from initial state distribution, $p_0(\mathbf{x}_0)$. We refer to this distribution as *q-distribution,* formally defined by,

$$q_{\mu,f,p_0}(\bar{\mathbf{x}},\bar{\mathbf{u}}) = p_0(\mathbf{x}_0)\prod_{k=0}^{N}\mu_k(\mathbf{u}_k \mid \mathbf{x}_k)\, f_k(\mathbf{x}_{k+1} \mid \mathbf{x}_k,\mathbf{u}_k)\,. \tag{3.6}$$

Therefore, the expected cost, in terms of q-distribution is given by,

$$\langle c(\bar{\mathbf{x}},\bar{\mathbf{u}})\rangle = \int_{\bar{\mathbf{x}},\bar{\mathbf{u}}} q_{\mu,f,p_0}(\bar{\mathbf{x}},\bar{\mathbf{u}})\, c(\bar{\mathbf{x}},\bar{\mathbf{u}})\, d\bar{\mathbf{x}}d\bar{\mathbf{u}}. \tag{3.7}$$

A q-distribution can be thought of as a probabilistic tube, comprised of all possible state-control trajectories generated by applying a certain policy to a given plant dynamics, at a given initial state distribution. Fig. 3.2 illustrates a distribution over state and control trajectories for a single dimension problem.

Having defined stochastic policies, since our goal is to address more general problems than the standard SOC, the next step is formulation of these problems.

3.3 Robust, Optimistic and Standard SOC Problems

As promised in Chaps. 1 and 2, the robustness issues of standard SOC are also to be addressed in our approach. To add even more flexibility and generality, our approach also handles a related problem, known as the *optimistic* (or risk-seeking) control [82, 83] to be defined shortly. Therefore, in this section, three variations of SOC problems are defined: Robust SOC, standard SOC and optimistic SOC. Although these problems are formulated separately in this section, later on in Sec. 3.6, it will be shown that a single solution can address all three.

3.3.1 Addressing Robustness: Pessimistic Control

As stated in Chaps. 1 and 2, one of the natural shortcomings of SOC is that, by assumption, it fully trusts the available model of the plant. Therefore, the optimal policy is specifically optimal for this nominal model. However, this optimality may be very sensitive to the model; if the *true* model of the system is slightly different from the available one, standard SOC may result in severe sub-optimality [72, 76, 84, 85]. There are several works in the literature which try to address the robustness issue. As mentioned in the previous chapter, a straightforward treatment of robustness results in RDP, which is similar to DP, except with additional dimensions and computational complexity compared to standard DP [75, 76]. Therefore, RDP is only applicable when the standard DP is tractable. Since for problems of our interest standard DP is intractable, existing works in RDP become useless.

In this chapter, robustness is addressed similar to the Hansen and Sargent's approach, described in their seminal text in economics[60] . This approach tries to robustify the control against misspecification of the stochastic model, by not fully trusting a given nominal model. Instead, this approach assumes a *set* of possible stochastic models for the plant, called the *ambiguity set*, and finds the optimal policy for the *worst-case* model within this set. It should be noted that, although the high-level definition and formulation of robustness is the same as Hansen's, our solution applies to general sequential problems with nonlinear and non-quadratic costs, while Hansen and Sargent provide the solution for special problems in economics or for a restricted class of problems with certain assumptions about the dynamics and uncertainties

To formally define the ambiguity set, suppose a nominal model of the plant is given by,

$$f\left(\mathbf{x}_{k+1} \mid \mathbf{x}_k, \mathbf{u}_k, k\right), \tag{3.8}$$

which we refer to as the *nominal dynamics* of the plant. The ambiguity set is a set, $\mathcal{A}$, which contains the nominal dynamics $f \in \mathcal{A}$, along with other possible dynamics. It is assumed that the (unknown) *true* dynamics is also contained within this set. The *robust policy*, $\mu^{\star r}$, is the policy which minimizes the expected cost for the *worst-case* dynamics within the ambiguity set. That is,

$$\mu^{\star r} \quad = \quad \underset{\mu \in \Pi_s}{\arg\min} \max_{f' \in \mathcal{A}} \left\langle c\left(\bar{\mathbf{x}}, \bar{\mathbf{u}}\right)\right\rangle .$$

Here, our focus is on ambiguity sets of a special form. Specifically, Kullback–Leibler (KL) bounds are used to define a bounded set of dynamics[1]. KL-divergence (or relative entropy) is formally defined in Appx. A.1. However, for our purposes in this chapter, KL-divergence can be simply considered as a measure of *distance* between two probability distributions. For the two distributions, $p\left(x\right)$ and $q\left(x\right)$, $\mathrm{KL}\left(p\left(x\right) \parallel q\left(x\right)\right)$ is a non-negative real number, with

[1]For other possible structures of ambiguity set see [85].

zero occurring only when $p(x) = q(x)$. Also, roughly speaking, the more different $p(x)$ looks from $q(x)$, the higher their KL-divergence. If the nominal dynamics is f, one can define a special form of ambiguity set given by,

$$\mathcal{A} = \left\{ f'(\mathbf{x}_{k+1} \mid \mathbf{x}_k, \mathbf{u}_k, k) \mid \mathrm{KL}\left(f'(\mathbf{x}_{k+1} \mid \mathbf{x}_k, \mathbf{u}_k, k) \parallel f(\mathbf{x}_{k+1} \mid \mathbf{x}_k, \mathbf{u}_k, k) \right) \leq \varepsilon_{f_k} \right\}. \quad (3.9)$$

where ε_{f_k} is a positive real constant. This set contains all dynamics which are not farther than ε_{f_k} from the nominal dynamics. This set obviously includes the nominal dynamics, f, because $\mathrm{KL}(f \parallel f) = 0 < \varepsilon_{f_k}$. Moreover, it's assumed that ε_{f_k} is chosen big enough so that the unknown true dynamics of the plant also falls within this set. The set defined by Eq. 3.9 is sometimes referred to as a *KL-ball*, with center, f, and radius, ε_{f_k}. For KL-ball ambiguity sets, the SOC problem, robustified against misspecification of the dynamics becomes:

$$\mu^{\star r} = \arg\min_{\mu \in \Pi_s} \max_{f'} \langle c(\bar{\mathbf{x}}, \bar{\mathbf{u}}) \rangle, \quad (3.10)$$

$$\textit{subject to} \quad \mathrm{KL}\left(f' \parallel f \right) \leq \varepsilon_{f_k}. \quad (3.11)$$

The problem above is reminiscent of other *minimax* problems in robust control, with the exception that it uses a stochastic definition for dynamics, instead of a deterministic one. The solution to the problem above is robust to misspecification of the dynamics of the plant. Similarly, SOC can also be robustified against the misspecification of the *initial distribution* of the state, $p_0(\mathbf{x}_0)$. In fact, in standard SOC, one assumes that the given initial state distribution is to be trusted completely. Therefore, using similar KL-ball ambiguity set assumptions on the initial state distribution, one has the robust SOC problem:

Problem 1. Find the robust policy, $\mu^{\star r}$, which minimizes the expected cost for the *worst-case* dynamics and initial state distribution within the ambiguity set.

$$\mu^{\star r} = \arg\min_{\mu \in \Pi_s} \max_{f', p_0'} \langle c(\bar{\mathbf{x}}, \bar{\mathbf{u}}) \rangle, \quad (3.12)$$

$$\textit{subject to} \quad \mathrm{KL}\left(f' \parallel f \right) \leq \varepsilon_{f_k}, \quad (3.13)$$

$$\textit{and} \quad \mathrm{KL}\left(p_0' \parallel p_0 \right) \leq \varepsilon_{p_0}. \quad (3.14)$$

The solution to the problem above, along with two other upcoming problems, will be developed in Sec. 3.6.

3.3.2 Optimism: Risk-seeking Control

The pessimistic problem defined previously makes a worst-case assumption about the true dynamics and the initial distribution of the system, which are assumed to be contained

within KL-balls centered on the nominal dynamics and the given initial distribution. If instead of a worst-case assumption, one makes a *best-case* assumption, the *optimistic control* (or risk-seeking control) problem is defined. In other words, if the max operation in Prob. 1 is replaced with a min operation, one has the optimistic control problem as follows:

Problem 2. Find the optimistic policy, $\mu^{\star o}$, which minimizes the expected cost for the *best-case* dynamics within the ambiguity set. That is,

$$\begin{aligned} \mu^{\star r} &= \underset{\mu\in\Pi_s}{\arg\min} \min_{f',p'_0} \left\langle c\left(\bar{\mathbf{x}},\bar{\mathbf{u}}\right)\right\rangle, && (3.15)\\ \textit{subject to} &\quad \mathrm{KL}\left(f' \parallel f\right) \leq \varepsilon_{f_k}, && (3.16)\\ \textit{and} &\quad \mathrm{KL}\left(p'_0 \parallel p\right) \leq \varepsilon_{p_0}. && (3.17) \end{aligned}$$

The optimism in these problems are justified by an "optimism in face of uncertainty" heuristic, which has been shown to be a useful heuristic for some problems [82, 83]. Specifically, for partially observable problems, a risk-seeking behavior compared to a greedy and optimal behavior may result in more informative observations. Moreover, it will be shown that risk-seeking control constitutes one extreme of our approach while the other extreme is robust control. The neutral case in the middle is the standard SOC. Therefore, although for our purposes, the standard and robust problems are more interesting, for the sake of unification the risk-seeking case is also considered in this chapter. Moreover, it will be shown that it can be easily handled with the same solution that applies to robust and standard SOC.

3.3.3 Standard SOC

If the ambiguity set is reduced to the nominal dynamics and initial state distribution, either of the two problems discussed previously in this section reduce to the standard SOC problem. In other words, for ambiguity sets defined as KL-balls, when the radii, ε_{f_k} and ε_{p_0} approach zero, the KL-balls shrink down and in the limit only contain the nominal dynamics and initial state distribution. Therefore, the optimistic problem can be reduced to standard SOC as follows:

Problem 3. Find the optimal policy, $\mu^{\star}$, which minimizes the expected cost for the nominal dynamic. That is,

$$\begin{aligned} \mu^{\star} &= \underset{\mu\in\Pi_s}{\arg\min} \operatorname*{ext}_{f',p'_0} \left\langle c\left(\bar{\mathbf{x}},\bar{\mathbf{u}}\right)\right\rangle, && (3.18)\\ \textit{subject to} &\quad \mathrm{KL}\left(f' \parallel f\right) \leq \varepsilon_{f_k}, && (3.19)\\ \textit{and} &\quad \mathrm{KL}\left(p'_0 \parallel p\right) \leq \varepsilon_{p_0}, && (3.20)\\ \varepsilon_{f_k} &\rightarrow 0, && (3.21)\\ \varepsilon_{p_0} &\rightarrow 0. && (3.22) \end{aligned}$$

This means that if one has the solution to either the optimistic or pessimistic optimal problems, the standard SOC solution can be recovered at the limit of $\varepsilon_{f_k}, \varepsilon_{p_0} \to 0$.

The three problems defined in this section are all variational problems, because they are optimization problems which seek optimal (conditional) distribution *functions*. In the robust problem, one has maximization and minimization sub-problems while the optimistic case is composed only of minimization sub-problems. All of this sub-problems are essentially problems of finding a distribution under which the expected value of a cost function is minimized (or maximized). Therefore, to be able to solve these three variations of SOC, one needs to know how to approach problems of *optimal distributions.*

3.4 Optimal Distributions

Previously, three related SOC problems were formulated, without discussing their solutions. The reason is that all of these variational problems are that of finding constrained or unconstrained *optimal distributions* (OD). Therefore, before approaching Probs. 1–3, one needs to know how to approach basic constrained and unconstrained OD problems. A discussion on OD problems constitutes the topic of this section.

3.4.1 Unconstrained Optimal Distribution Problems

Let us define a simple unconstrained OD problem.

Problem 4. Find the distribution $\Pr(\mathbf{X} = \mathbf{x}) = p(\mathbf{x})$ such that the expected value of the cost function, $c(\mathbf{x})$ under $p(\mathbf{x})$ is minimized.

$$p^{\star}(\mathbf{x}) = \arg\min_{p(\mathbf{x})} \langle c(\mathbf{x}) \rangle_{p(\mathbf{x})} \tag{3.23}$$

$$= \arg\min_{p(\mathbf{x})} \int_{\mathbf{x}} c(\mathbf{x})\, p(\mathbf{x})\, d\mathbf{x}. \tag{3.24}$$

These problems and their solutions are frequent topics in stochastic optimization methodologies, such as *probability collectives* [86, 87].

A Proximal Point Method and KL-Penalized Problems

The *proximal point* (PP) method was first developed by Rockafellar in [79] with some complete reviews available in [78, 88, 89]. However, here the discussion is limited to relevant high-level concepts. The basic idea in PP algorithms is that instead of solving the minimization in one step, it might be easier to gradually revise and refine an initial solution, by finding an improved solution in the "proximity" of the one before. Thus, the general form of proximal algorithms is,

$$f^{(i+1)} = \arg\min_{f} \left\{ cost\left(f\right) + \beta_i \mathrm{D}\left(f \parallel f^{(i)}\right) \right\}, \tag{3.25}$$

where f is the argument of optimization and $\mathrm{D}\left(\cdot \parallel \cdot\right)$, the proximity functional, is a distance-like measure of the proximity of the solutions. The proximity functional penalizes the deviation of the solution from the previous solution, and therefore, keeps the solution in the *proximity* of the one before. For the choice of proximity function, originally, quadratic distances have been used [79]. However, very general types of distance-like functions have since been introduced, including entropy-like distance measures [78, 89], which are particularly useful in variational inference problems. What follows is the PP problem, which uses a relative entropy (or KL-divergence) term as the proximity function.

Problem 5. Iteratively solve,

$$p^{(i+1)} \quad = \quad \arg\min_{p} \left\{ \left\langle c\left(\mathbf{x}\right)\right\rangle_{p(\mathbf{x})} + \beta_i \mathrm{KL}\left(p \parallel p^{(i)}\right) \right\} \tag{3.26}$$

Each iteration of the above is a variational problem penalized by KL-divergence (or relative entropy) penalty function. We refer to this problem as a *KL-penalized* variational problem. The reason that the above PP problem is preferred to the original problem is that the original problem is linear in $p\left(\mathbf{x}\right)$ while the KL-penalized problem is strictly convex [2], and therefore, finding the solution to the PP problem is easier. In fact, the solution to an iteration of the PP problem above can be found in *closed-form* and is given by the following lemma.

Lemma 1. *The solution to the KL-penalized problem,*

$$p^{\star(\beta,p_0)} \quad = \quad \min_{p} \left\{ \left\langle c\left(\mathbf{x}\right)\right\rangle_{p(\mathbf{x})} + \beta KL\left(p \parallel p_0\right) \right\}, \tag{3.27}$$

is the Boltzmann distribution (with degeneracy, p_0),

$$p^{\star(\beta,p_0)} = \frac{1}{Z\left(\beta\right)} p_0\left(\mathbf{x}\right) e^{-\frac{c(\mathbf{x})}{\beta}}, \tag{3.28}$$

with the partition function,

$$Z\left(\beta\right) = \int_{\mathbf{x}} p_0\left(\mathbf{x}\right) e^{-\frac{c(\mathbf{x})}{\beta}} d\mathbf{x}. \tag{3.29}$$

Proof. See [90] □

[2] It is strictly convex in ${}^{p(\mathbf{x})}/_{p^{(i)}(\mathbf{x})}$.

Lemma 1 is very useful because KL-penalized problems appear frequently in our treatment, as well as in most works dealing with approximate inference-based solutions of SOC. In Sec. 3.8, a review on related works which use this lemma will be given.

Prob. 5 is an unconstrained problem with a relative entropy penalty term in the cost function. Because of our assumption on the form of the ambiguity set in Sec. 3.3, *constrained* OD problems with KL constraints also appear in our work.

3.4.2 Constrained Optimal Distribution Problems

So far, it was shown that (unconstrained) OD problems can be solved iteratively using a PP method, and that each iteration of the PP method is a KL-penalized problem, which in turn, admits a closed-from solution given by Lemma 1. Here, the solution to *constrained* OD problems is reviewed. In particular, the focus is on constrained OD problems, where the constraints are KL-bounds. We refer to the problems as *KL-constrained* problems. KL-constrained problems show up in our robust and optimistic SOC problems defined in Sec. 3.3, because the ambiguity sets were defined by KL-bounds. It will be shown that the solution to KL-constrained problems turn out to be closely related to the solution to KL-penalized problems seen earlier. The KL-constrained problem is formally defined as follows:

Problem 6. Find the solution to the KL-constrained problem below.

$$p^{*(\varepsilon,p_0)}(\mathbf{x}) = \arg\min_{p(\mathbf{x})} \langle c(\mathbf{x})\rangle_{p(\mathbf{x})}, \tag{3.30}$$

$$\textit{subject to} \qquad \mathrm{KL}(p(\mathbf{x}) \parallel p_0(\mathbf{x})) \leq \varepsilon. \tag{3.31}$$

Prob. 6 is essentially a variational problem on a KL-ball. The KL constraint can be enforced using a Lagrange multiplier method. For an in-depth introduction to the Lagrange multiplier method see [91]. The Lagrangian reads,

$$\mathcal{L}(p(\mathbf{x}), \beta) = \langle c(\mathbf{x})\rangle_{p(\mathbf{x})} + \beta(\mathrm{KL}(p(\mathbf{x}) \parallel p_0(\mathbf{x})) - \varepsilon), \tag{3.32}$$

with $\beta \geq 0$ as the *Lagrange multiplier*. Therefore, problem 6 is equivalent to:

$$\min_{p(\mathbf{x})} \min_{\beta} \mathcal{L}(p(\mathbf{x}), \beta). \tag{3.33}$$

The Lagrangian dual is formed by changing the order of minimization. Thus, the dual problem is,

$$\min_{\beta} \min_{p(\mathbf{x})} \mathcal{L}(p(\mathbf{x}), \beta). \tag{3.34}$$

Since Prob. 6 is strictly convex[3], strong duality holds and the solutions to the primal

[3]It is convex in $p(\mathbf{x})/p_0(\mathbf{x})$.

and dual problems are equivalent . Therefore, one can continue with the dual problem. The inner minimization in the dual is,

$$\mathcal{L}\left(p^{\star(\beta,p_0)},\beta\right) = \min_{p(\mathbf{x})}\left\{\langle c(\mathbf{x})\rangle_{p(\mathbf{x})} + \beta \mathrm{KL}\left(p(\mathbf{x}) \parallel p_0(\mathbf{x})\right) - \beta\varepsilon\right\}, \tag{3.35}$$

$$\text{for} \quad \beta \geq 0 \tag{3.36}$$

where the last term in the r.h.s is constant w.r.t. $p(\mathbf{x})$ and, therefore, can be dropped from the inner minimization problem. As for the Lagrangian multiplier,β, the case where $\beta = 0$ happens only when the solution is inside the KL-ball and $\beta > 0$ when it's on the boundary, $\mathrm{KL}\left(p(\mathbf{x}) \parallel p_0(\mathbf{x})\right) = \varepsilon$. Here, the case when $\beta = 0$ is dismissed. The reason is that unless $p_0(\mathbf{x})$ is the optimal Dirac distribution itself, the solution to the problem above always lies on the boundary. Therefore, for all non-trivial problems, it is always the case that $\beta > 0$. By taking $\beta > 0$, it becomes evident that the inner minimization problem is a KL-penalized problem and, therefore, Lemma 1 applies. Hence,

$$p^{\star(\beta,p_0)} = \frac{1}{Z(\beta)} p_0(\mathbf{x})\, e^{-\frac{c(\mathbf{x})}{\beta}}. \tag{3.37}$$

Plugging the solution above back into the dual problem in Eq. 3.34 yields,

$$\beta^\star(\varepsilon) = \arg\min_{\beta>0}\left[-\beta \log Z(\beta) - \beta\varepsilon\right]. \tag{3.38}$$

For our purposes, the solution of Eq. 3.38 is not as important as the general relationship between ε and $\beta^\star(\varepsilon)$, and specially the relationship at the limits of zero and infinity. This relationship is interesting, because it states that a KL-constrained problem with radius ε is equivalent to a KL-penalized problem with a penalty coefficient of $\beta^\star(\varepsilon)$ and vice versa. Ultimately, this relationship establishes a two-way relationship between KL-penalized and KL-constrained problems. This relationship has also been studied in the context of economics, in [92, 93].

3.4.3 The Relationship Between KL-Constrained and KL-Penalized Problems

To investigate the general properties of the relationship between KL-constrained and KL-penalized problems, one can observe from Eq. 3.38 that $\partial\beta^\star/\partial\varepsilon$ is always negative (since $\beta > 0$). Therefore, if the solution, $\beta^\star$, exists, it is decreasing in ε. This suggests an interesting (two-way) relationship for Probs. 5 and 6: The solution to problem 5 with $\beta^\star(\varepsilon)$ is a solution to problem 6 with ε, and *vice versa*. This means that adding a relative entropy term to the objective function has the same effect as restricting the problem to a KL-ball. Thus, an (unconstrained) variational problem with a relative entropy term in the objective is equivalent to a variational problem without the relative entropy term, but restricted to a

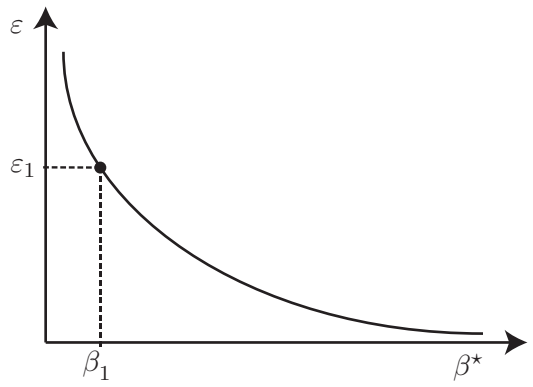


Figure 3.3 – The relationship between ε and $\beta^\star$, which states that Prob. 6 with ε_1 is equivalent to Prob. 5 with β_1.

KL-ball with an appropriate radius.

In mathematical terms,

$$p^{*(\varepsilon,p_0)}(\mathbf{x}) = p^{\star(\beta^\star(\varepsilon),p_0)}, \tag{3.39}$$

where $\beta^\star$ and ε are related by Eq. 3.38 [4]. The radius of the KL-ball and the scaling coefficient of the relative entropy (or the *temperature*[5]) are related by Eq. 3.38, and because of the monotonicity of the mapping, it is the case that the lower the radius ,ε, the higher the corresponding temperature, $\beta^\star$, and vice versa. In particular, for $\varepsilon \to 0$ one has $\beta^\star \to +\infty$, and for $\varepsilon \to +\infty$ one has $\beta^\star \to 0$. To give the reader an image, the conceptual relationship between ε and $\beta^\star$ appears in Fig. 3.3, keeping in mind that the exact relationship is a complicated functional of $c(\mathbf{x})$ and $p_0(\mathbf{x})$.

These results were anticipated because the solution of Prob. 6 with the KL-ball constraint should approach the optimal unconstrained solution when the radius of the KL-ball is increased infinitely. Also, when the radius of the KL-ball approaches 0, the set of distributions satisfying the constrained shrinks to $\{p_0(\mathbf{x})\}$, which means that the solution is bound to approach $p_0(\mathbf{x})$. For the KL-penalty term in Prob. 5, when the penalty coefficient, β, approaches zero, the unconstrained optimal solution is recovered and when penalty coefficient increases, the KL-penalty dominates and holds the solution closer to $p_0(\mathbf{x})$.

So far, the unconstrained and constrained minimization problems were studied. In our treatments, we also face constrained *maximization* problems and so, a review on these problems will be given shortly.

[4] One could arrive at the same statement much quicker by simply pointing out that a KL-penalized problem is the Lagrangian relaxation of a KL-constrained problem. However, our discussion explicitly relates KL bound, ε , and the corresponding KL-penalty coefficient, $\beta^\star$.

[5] The coefficient scaling relative entropy or entropy terms are historically referred to as temperature, because they first appeared in statistical physical problem.

3.4.4 The KL-Constrained Maximization Problem

The robust SOC defined in Sec. 3.3 is a minimax problem, where the max part requires finding a *maximizing* distribution on a KL-ball ambiguity set. Therefore, it is useful to review the problem of maximizing the expected cost on a KL-ball. It is easy to see that in order to find the maximizing solution one can simply negate the cost function and, once again, perform a minimization. That is,

$$\arg\max_{p(\mathbf{x})} \langle c(\mathbf{x})\rangle_{p(\mathbf{x})} = \arg\min_{p(\mathbf{x})} \langle -c(\mathbf{x})\rangle . \tag{3.40}$$

Repeating the same steps as before, with the Lagrange multiplier,$\beta' \geq 0$, one has,

$$p(\mathbf{x})^{\star(\beta',p_0)} = \frac{1}{Z(\beta')} p_0(\mathbf{x})\, e^{+\frac{c(\mathbf{x})}{\beta'}}, \tag{3.41}$$

$$Z\left(\beta'\right) = \int_{\mathbf{x}} p_0(\mathbf{x})\, e^{+\frac{c(\mathbf{x})}{\beta'}} d\mathbf{x}, \tag{3.42}$$

and,

$$\beta'^{\star} = \arg\min_{\beta'\geq 0} \left[-\beta' \log Z\left(\beta'\right) - \beta'\varepsilon\right]. \tag{3.43}$$

Comparing the maximization problem with the minimization counterpart, one can easily realize that switching the sign of the Lagrange multiplier in one of the problems results in the other; the reader can simply replace $\beta' = -\beta$ in the above to arrive at the same expressions as in Eq. 3.38. It should be noted that, while the partition function, $Z(\beta) = \int_{\mathbf{x}} p_0(\mathbf{x})\, e^{-\frac{c(\mathbf{x})}{\beta}} d\mathbf{x}$, for the minimization problem was always well-defined (assuming $c(\mathbf{x}) \geq 0$, $c(\mathbf{x}) \neq cte.$ and $\beta > 0$), the partition function in Eq. 3.42 for the maximization problem may not always be finite. Therefore, for the Eq. 3.42, the range of β where the integral is finite may be bounded. It's helpful to briefly find the admissible range of β a simple example:

Example 1. Let's consider a scalar problem with a quadratic cost $c(x) = ax^2$ and a nominal Gaussian distribution $p_0(x) = N\left(0, \sigma^2\right)$. For this problem, the integrand in Eq. 3.42 is proportional to, $e^{\left(\frac{ax^2}{\beta'} - \frac{x^2}{\sigma^2}\right)}$, which suggests that the partition function is well-defined for $a\sigma^2 < \beta'$(or, equivalently, $\beta < -a\sigma^2$).

This section is concluded with a remark summarizing the results for constrained OD problems:

Remark 1. The constrained maximization and minimization problems may be combined into a single *extremization* problem, where the solution given by Lemma 1 is a maximizing solution when $\beta > 0$ and it's a minimizing solution when $-\infty < \beta < \beta_{limit}$. In other words, the solution to,

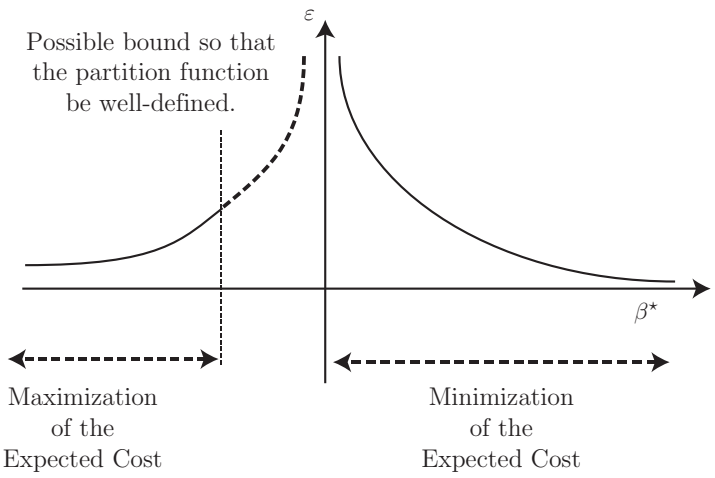


Figure 3.4 – The relationship between ε and $\beta^\star$, including negative values for $\beta^\star$.

$$\arg\operatorname{ext}_{p(\mathbf{x})} \langle c(\mathbf{x})\rangle_{p(\mathbf{x})}, \tag{3.44}$$

$$\textit{subject to} \qquad \mathrm{KL}\left(p(\mathbf{x}) \parallel p_0(\mathbf{x})\right) \leq \varepsilon, \tag{3.45}$$

given by Eq. 3.28 is equivalent to the solution of the KL-penalized problem,

$$\arg\min \left\{ \langle c(\mathbf{x})\rangle_{p(\mathbf{x})} + \beta^\star(\varepsilon)\, \mathrm{KL}\left(p \parallel p_0\right) \right\}, \tag{3.46}$$

when $\beta^\star$ is positive, and to,

$$\arg\max \left\{ \langle c(\mathbf{x})\rangle_{p(\mathbf{x})} + \beta^\star(\varepsilon)\, \mathrm{KL}\left(p \parallel p_0\right) \right\}, \tag{3.47}$$

when $\beta^\star$ is negative. Fig. 3.4 depicts the relationship between KL-constrained minimization/maximization problems and KL-penalized problems.

3.4.5 Relationship to Inference

In this section some important optimal distribution problems were reviewed. However, their relation to inference was not discussed. Indeed, inference problems such as, finding the posterior distribution given a prior distribution and some evidence, can be formulated as variational problems. For instance, the posterior can be found as the optimal distribution which minimizes a *free energy* functional. Such variational formulations of Bayesian inference are known as variational Bayes methods [94, 95]. Space doesn't permit a complete presentation of variational inference, nor is it really necessary. Instead, it's easier to simply point out that the solution to the KL-penalized and KL-constrained problems above, given by Boltzmann distribution in Eq. 3.28, can be interpreted as Bayesian posterior calculation.

Specifically, it's easy to see that:

Remark 2. In Eq. 3.28, if $p_0(\mathbf{x})$ is a prior distribution and $-\beta^{-1}c(\mathbf{x})$ is the log-likelihood of the evidence given $\mathbf{x}$, then $p^{\star\beta}(x)$ is the posterior distribution.

The above can be simply verified by remembering that the Bayes law states that, $posterior = {}^{prior \times likelihood}/_{normalization\, constant}$. This observation ultimately enables us to solve OD problems using any approximate method for posterior calculation. Indeed, the solution to the KL-penalized problems is equal to the solution of the following Bayesian posterior problem:

Problem 7. A prior distribution is given by $p_0(\mathbf{x})$. R is binary random variable and it is observed as $R = 1$. Also, it is known that,

$$\Pr(R = 1 \mid \mathbf{X} = \mathbf{x}) = e^{-\frac{c(\mathbf{x})}{\beta}}. \tag{3.48}$$

given the information that $R = 1$, the posterior distribution is simply given by,

$$\frac{1}{Z(\beta)} p_0(\mathbf{x})\, e^{-\frac{c(\mathbf{x})}{\beta}}. \tag{3.49}$$

Because of the equivalence of the above inference problem and the KL-penalized problems seen above, whenever a cost function $c(\mathbf{x})$ is to be minimized under the unknown distribution, one can define an auxiliary random variable and assume it is observed to be in a state with a log-likelihood proportional to $-\beta^{-1}c(\mathbf{x})$, and solve the above posterior problem. This relationship between cost (or utility) and log-likelihood is well-known and studied in detail in [57, 58]. Similar to [16, 28] an auxiliary binary random variable, R, is defined such that,

$$\Pr(R = 1 \mid \mathbf{X} = \mathbf{x}) = e^{-\frac{c(\mathbf{x})}{\eta}}. \tag{3.50}$$

The equation above is, indeed, a Boltzmann distribution for a system with energy function, $E(\mathbf{x}) = c(\mathbf{x})$, at temperature η. With these definitions, solving the KL-penalized minimization problem (Prob. 5) with the penalty coefficient β, (or KL-ball constrained Prob. 6 with the radius $\varepsilon(\beta)$) is equivalent to a standard posterior Bayesian having observed $R = 1$, when $\Pr(R = 1 \mid \mathbf{X} = \mathbf{x}) = e^{-\frac{c(\mathbf{x})}{\eta}}\Big|_{\eta=\beta}$. This relation is depicted in Fig. 3.5. It is clear from the figure that the expected cost under the posterior is lower than the expected cost under the prior. The KL-penalty coefficient, β, controls how strength of the evidence; when $\beta \to 0$ the evidence becomes sharp at the minimum of the cost function and, as a result, posterior will also be strongly shifted towards the region where the cost function is minimum.

Thus, the solution to the KL-penalized and KL-constrained problems above can be thought of as *special* forms of Bayesian posterior inference, where the log-likelihood function is scaled by a parameter ${}^{\beta}/_{\eta}$ (or the temperature is scaled by β). It will be shown that the fact that the solutions to KL problems are equal to simple posterior calculation allows us to apply approximate inference methods to solve SOC.

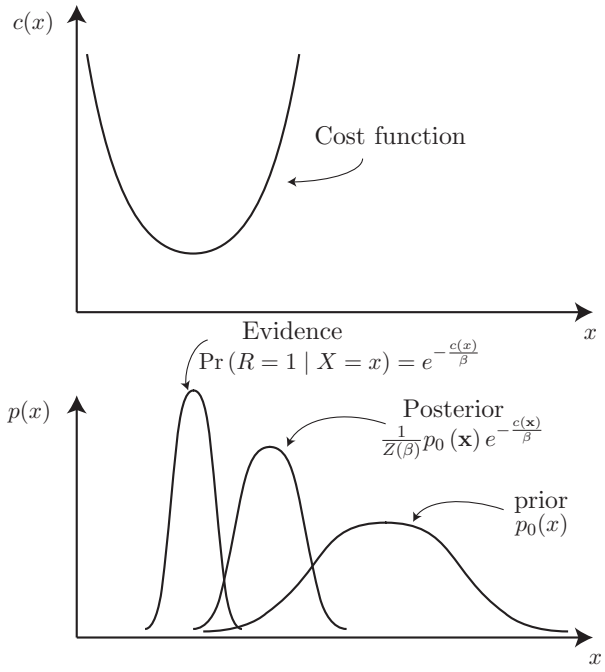


Figure 3.5 – Inference interpretation of Eq. 3.28. A KL-penalized or a KL-constrained problem can be interpreted as standard posterior calculation.

3.5 A Proximal Point Method for SOC

In this section, as a first step, the three variations of SOC defined in Sec. 3.3 are combined into a unified problem. This problem is then transformed into a KL-penalized problem and solved using a PP method. To begin, let us present the unified SOC problem:

Problem 8. Find the optimal policy,

$$\begin{aligned} \mu^\star \quad &= \quad \arg\min_{\mu\in\Pi_s} \operatorname*{ext}_{f'} \left\langle c\left(\bar{\mathbf{x}}, \bar{\mathbf{u}}\right)\right\rangle, \\ \textit{subject to} \quad & \quad \mathrm{KL}\left(f' \parallel f\right) \leq \varepsilon_{f_k}, \\ \textit{and} \quad & \quad \mathrm{KL}\left(p_0' \parallel p_0\right) \leq \varepsilon_{p_0}, \end{aligned} \tag{3.51}$$

where the reader can easily verify that, depending on whether the extremum is a minimum or maximum, the optimistic and pessimistic variants are resulted respectively, and when $\varepsilon_{f_k} \to 0$, the standard SOC is recovered.

The problem above is a KL-constrained problem. According to the results n Sec. 3.4.3 which were summarized by Remark 1, the KL-constrained problems above can be converted to the following equivalent KL-penalized problem:

Problem 9. Find the optimal policy,

$$\begin{aligned}\mu^\star &= \arg\min_{\mu\in\Pi_s}\Big\{\operatorname*{ext}_{f'}\left\{\langle c\left(\bar{\mathbf{x}},\bar{\mathbf{u}}\right)\rangle\right.\\ &+ \sum_{k=0}^{N}\beta_k\mathrm{KL}\left(f'\parallel f\right)\\ &+ \left.\gamma_0\mathrm{KL}\left(p_0'\parallel p_0\right)\right\}\Big\},\end{aligned}\tag{3.52}$$

where each pair of $(\beta_k, \varepsilon_{f_k})$ and $(\gamma_0, \varepsilon_{p_0})$ are related according to the duality results described in Sec. 3.4.3 and depicted in Fig. 3.4.

The problem above is now an unconstrained OD problem. In Sec. 3.4 it was shown that such unconstrained problems can be solved using a PP method. Therefore, by adding a proximity function to the above problem, the PP problem to be solved iteratively can be formulated as follow:

Problem 10. Iteratively solve for policies,

$$\begin{aligned}\mu^{(i+1)} &= \arg\min_{\mu\in\Pi_s}\Big\{\operatorname*{ext}_{f,'p_0'}\left\{\langle c\left(\bar{\mathbf{x}},\bar{\mathbf{u}}\right)\rangle +\right.\\ &+ \sum_{k=0}^{N}\beta_k\mathrm{KL}\left(f'\parallel f\right)+\gamma_0\mathrm{KL}\left(p_0'\parallel p_0\right)\\ &+ \left.\zeta^{(i)}\mathrm{D}_\pi\left(\mu\parallel\mu^{(i)}\right)\right\}\Big\}.\end{aligned}\tag{3.53}$$

The proximity function of our choice is a weighted sum of (conditional) KL divergences, where each term penalizes the divergence of $\mu_k^{(i+1)}$ from $\mu_k^{(i)}$, i.e.,

$$\begin{aligned}\zeta_i\mathrm{D}_\pi\left(\mu\parallel\mu^{(i)}\right) &= \sum_{k=0}^{N}\left[\zeta_i\alpha_k'\mathrm{KL}\left(\mu_k\parallel\mu_k^{(i)}\right)\right]\\ &= \sum_{k=0}^{N}\alpha_k\mathrm{KL}\left(\mu_k\parallel\mu_k^{(i)}\right).\end{aligned}\tag{3.54}$$

In the above, $\zeta^{(i)}\alpha_k'$ was renamed to α_k. Therefore, the PP iteration consists in solving,

$$\begin{aligned}\mu^{(i+1)} &= \arg\min_{\mu\in\Pi_s}\Big\{\operatorname*{ext}_{f,'p_0'}\left\{\langle c\left(\bar{\mathbf{x}},\bar{\mathbf{u}}\right)\rangle +\right.\\ &+ \sum_{k=0}^{N}\beta_k\mathrm{KL}\left(f'\parallel f\right)+\gamma_0\mathrm{KL}\left(p_0'\parallel p_0\right)\\ &+ \left.\sum_{k=0}^{N}\alpha_k\mathrm{KL}\left(\mu_k\parallel\mu_k^{(i)}\right)\right\}\Big\}.\end{aligned}\tag{3.55}$$

The solution to the PP problem above is given in the next section.

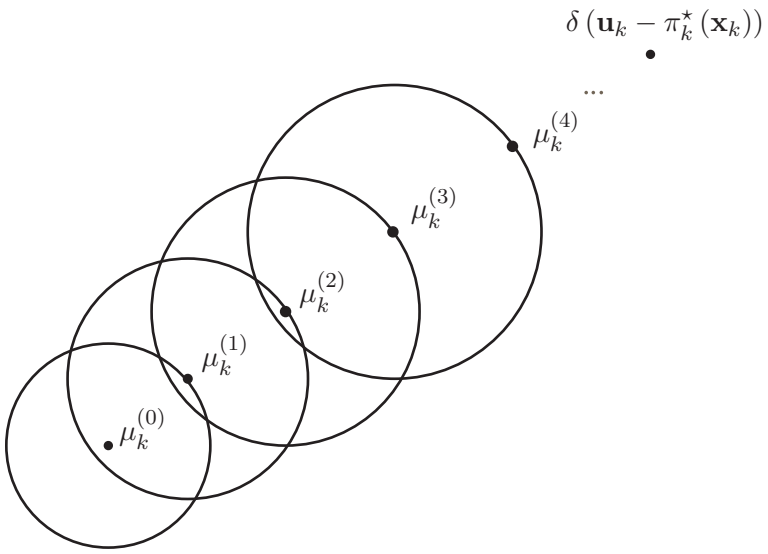


Figure 3.6 – Iterations of proximal policy solutions for time slice k. In each update of the policy at time k, a better policy on a KL-ball of the one before is found.

3.6 The Solution to the Proximal Point Problem

There exists still at least two ways that Prob. 10 can be solved, which are discussed in this section. First, it will be shown that Prob. 10 can be broken into sub-problems which are individually of the same form as the basic problem Sec. 3.4 and, therefore, admit to a closed-form solution given by Lemma 1. The result is a DP algorithm which updates the policy in a backward sweep. We refer to this backward recursion as *synchronous updates* of the policy. This backward recursion is similar to the original DP algorithm in Chap. 2, expect the exact minimizations are replaced by approximate soft-mins.

Secondly, it will be shown that although the problem is *jointly non-convex* in,

$$\left\{p_0', f_k', \mu_k \mid k = 0 : N\right\},$$

it is *separately* convex in each of the variables. That is, if one fixes all variables except one, the resulting problem is convex in this remaining variable. Therefore, a solution based on *alternating convex optimization* is possible. This roughly means that one can optimize each variable separately keeping the others fixed and alternate the fixed variables until convergence. (See [96] for more details on alternating convex optimization). This approach allows one to have *asynchronous updates*, i.e., as opposed to the backward sweep which requires a sequence of updates backward in time, one can update the policy at any time slice, μ_k, and in any order.

In both approaches, each update of the policy, results in an improved policy which is closer to the deterministic optimal policy. Fig. 3.6 depicts how the updates of the policy at time slice k improve towards the optimal policy.

3.6.1 Backward, Synchronous Updates

Let us expand the problem in each PP iteration in Prob. 10 in the following way, where the dependencies of all functions on $\mathbf{x}_k$ and $\mathbf{u}_k$ are suppressed for brevity.

$$\begin{aligned}
&\min_{\{\mu_k|k=0:N\}} \operatorname*{ext}_{\{p'_0,f'_k\}} \left\{ \int_{\bar{\mathbf{x}},\bar{\mathbf{u}}} \left(p'_0 \prod_{k'=0}^{N} \mu_k f'_k \right) c\left(\bar{\mathbf{x}},\bar{\mathbf{u}}\right) d\bar{\mathbf{x}} d\bar{\mathbf{u}} \right. \\
+ \quad & \gamma_0 \int_{\bar{\mathbf{x}},\bar{\mathbf{u}}} \left(p'_0 \prod_{k'=0}^{N} \mu_{k'} f'_{k'} \right) \log \frac{p'_0}{p_0} d\bar{\mathbf{x}} d\bar{\mathbf{u}} \\
+ \quad & \sum_{k=0}^{N} \alpha_k \int_{\bar{\mathbf{x}},\bar{\mathbf{u}}} \left(p'_0 \prod_{k'=0}^{N} \mu_{k'} f'_{k'} \right) \log \frac{\mu_k}{\mu_k^{(i)}} d\bar{\mathbf{x}} d\bar{\mathbf{u}} \\
+ \quad & \left. \sum_{k=0}^{N} \beta_k \int_{\bar{\mathbf{x}},\bar{\mathbf{u}}} \left(p'_0 \prod_{k'=0}^{N} \mu_{k'} f'_{k'} \right) \log \frac{f'_k}{f_k} d\bar{\mathbf{x}} d\bar{\mathbf{u}} \right\}. \qquad (3.56)
\end{aligned}$$

In the above, the log terms come from expanding the KL-divergence terms according to their definition in Appx. A.1. The terms in the last time step are then separated by moving the extremization and minimizations inside:

$$\begin{aligned}
&\min_{\{\mu'_k|k=0:N-1\}} \operatorname*{ext}_{\{p'_0,f'_k|k=0:N-1\}} \int_{\mathbf{x}_{0:N-1},\mathbf{u}_{0:N-1}} \left\{ \left(p'_0 \prod_{k'=0}^{N} \mu'_{k'} f'_{k'} \right) \sum_{k=0}^{N-1} \left(c_k + \beta_k \log \frac{f'_k}{f_k} + \alpha_k \log \frac{\mu'_k}{\mu_k^{(i)}} \right) \right. \\
&\left. + \operatorname*{ext}_{f'_{N-1}} \int_{\mathbf{x}_N} f'_{N-1} \Big\{ \beta_{N-1} \log \frac{f'_{N-1}}{f_{N-1}} + \min_{\mu'_N} \int_{\mathbf{u}_N} \mu'_N \big\{ c_N + \alpha_N \log \frac{\mu'_N}{\mu_N^{(i)}} \big\} d\mathbf{u}_N \Big\} d\mathbf{x}_N \right\} d\mathbf{x}_{0:N-1} d\mathbf{u}_{0:N-1}. \qquad (3.57)
\end{aligned}$$

The two inner-most problems in the equation above are unconstrained minimization problems penalized by a relative entropy term, which are exactly of the same form as Prob. 5. Specifically, the inner most problem is,

$$\begin{aligned}
&\min_{\mu'_N} \int_{\mathbf{u}_N} \mu'_N \left(c_N + \alpha_N \log \frac{\mu'_N}{\mu_N^{(i)}} \right) d\mathbf{u}_N, \qquad (3.58) \\
= \quad & \min_{\mu'_N} \left(\langle c_N \rangle_{\mu'_N} + \alpha_N \mathrm{KL}\left(\mu'_N \parallel \mu_N^{(i)} \right) \right). \qquad (3.59)
\end{aligned}$$

Therefore, the closed-form solutions given by Lemma 1 can be used. Thus, solving for μ'_N using Lemma 1,

$$\mu_N^{(i+1)}\left(\mathbf{u}_N \mid \mathbf{x}_N\right) = \frac{\mu_N^{(i)}\left(\mathbf{u}_N \mid \mathbf{x}_N\right) e^{-\frac{c_N(\mathbf{x}_N,\mathbf{u}_N)}{\alpha_N}}}{Z_{\mu_N}^{(i)}\left(\alpha_N, \mathbf{x}_N\right)}, \qquad (3.60)$$

and by plugging 3.60 into 3.59, one has the partition function given by,

$$Z_{\mu_N}^{(i+1)}\left(\alpha_N, \mathbf{x}_N\right) = \int_{\mathbf{u}_N} \mu_N^{(i)}\left(\mathbf{u}_N \mid \mathbf{x}_N\right) e^{-\frac{c_N(\mathbf{x}_N, \mathbf{u}_N)}{\alpha_N}} d\mathbf{u}_N. \tag{3.61}$$

Therefore, the second innermost optimization problem is becomes,

$$\begin{aligned} & \underset{f'_{N-1}}{\text{ext}} \int_{\mathbf{x}_N} f'_{N-1}\Big\{\beta_{N-1} \log \frac{f'_{N-1}}{f_{N-1}} + Z_{\mu_N}^{(i+1)}\left(\alpha_N, \mathbf{x}_N\right)\Big\} d\mathbf{x}_N, & (3.62) \\ = & \underset{f'_{N-1}}{\text{ext}} \left(\left\langle Z_{\mu_N}^{(i+1)} \right\rangle_{f'_{N-1}} + \beta_{N-1} \mathrm{KL}\left(f'_{N-1} \parallel f_{N-1}\right)\right). & (3.63) \end{aligned}$$

One can see that the above has the same form as the problem in Remark 1. Therefore, results of Remark 1 and Lemma 1 may be applied, which results in,

$$f_{N-1}^{(i+1)}\left(\mathbf{x}_N \mid \mathbf{x}_N, \mathbf{u}_{N-1}\right) = \frac{f_{N-1}\left(\mathbf{x}_N \mid \mathbf{x}_{N-1}, \mathbf{u}_{N-1}\right) e^{\frac{\alpha_N \log Z_{\mu_N}(\alpha_N, \mathbf{x}_N)}{\beta_{N-1}}}}{Z_{f_{N-1}}^{(i)}\left(\beta_{N-1}, \mathbf{x}_{k-1}, \mathbf{u}_{N-1}\right)}.$$

Again, plugging the above back into 3.63 results in,

$$Z_{f_{N-1}}^{(i)}\left(\beta_{N-1}, \mathbf{x}_{k-1}, \mathbf{u}_{N-1}\right) = \int_{\mathbf{x}_N} f_{N-1}\left(\mathbf{x}_N \mid \mathbf{x}_{N-1}, \mathbf{u}_{N-1}\right) e^{\frac{\alpha_N \log Z_{\mu_N}(\alpha_N, \mathbf{x}_N)}{\beta_{N-1}}} d\mathbf{x}_N. \tag{3.64}$$

One can continue factorizing the minimization and extremization, and solve using Lemma 1. This gives the general update equations as,

$$\boxed{f_k^{(i+1)} = \frac{f_k^{(i)} e^{\frac{\alpha_{k+1} \log Z_{\mu_{k+1}}(\alpha_{k+1})}{\beta_k}}}{Z_{f_k}^{(i)}\left(\beta_k\right)}.} \tag{3.65}$$

$$\boxed{\mu_k^{(i+1)} = \frac{\mu_k^{(i)} e^{-\frac{c_k - \beta_k \log Z_{f_k}(\beta_k)}{\alpha_k}}}{Z_{\mu_k}^{(i)}\left(\alpha_k\right)}.} \tag{3.66}$$

and,

$$\boxed{Z_{f_k}^{(i)}\left(\beta_k\right) = \int_{\mathbf{x}_{k+1}} f_k e^{\frac{\alpha_{k+1} \log Z_{\mu_{k+1}}(\alpha_{k+1})}{\beta_k}} d\mathbf{x}_{k+1}.} \tag{3.67}$$

$$\boxed{Z_{\mu_k}^{(i)}\left(\alpha_k\right) = \int_{\mathbf{u}_k} \mu_k^{(i)} e^{-\frac{c_k - \beta_k \log Z_{f_k}(\beta_k)}{\alpha_k}} d\mathbf{u}_k.} \tag{3.68}$$

These update equations constitute the core of our contribution.

3.6.2 Alternating Convex Optimization

Since Prob. 1 is separately convex in each of the variables $\{p'_0, f'_k, \mu_k \mid k = 0 : N\}$, one can apply the alternating convex optimization method [96]. The idea is to keep all variables fixed, except one, and perform the minimization w.r.t. this remaining variable. Then, the solution for this variable is used as its fixed value, and the problem is solved w.r.t. another variable. The variable is then "alternated" until convergence is resulted. Moreover, it is known that as long as a solution to the minimization exists, each iteration of the alternating minimization decreases the objective[6]. Assuming that the optimal objective is bounded from below, and in each step the objective is not increased, the convergence follows – albeit possibly to a local minimum [96].

To apply alternating convex method to the problem at hand, let us fix all variables except μ_k and minimize with respect to this variable. It is easy to see that update Eqs. 3.66 and 3.68 are the solution. Similarly, if one fixes all variables, except f'_k, the solution is given by Eqs. 3.65 and 3.67. Since in the alternating optimization method one can optimize the variables in any order, it is easy to see that this approach simply amounts to update Eqs. 3.65–3.68 applied to any time slice, instead of strictly updating them backwards in time, hence the name, *asynchronous*.

3.6.3 Viewing Updates as Messages

Referring back to update Eqs. 3.65–3.68, one can begin to interpret each update as a special type of message in a message passing algorithm. Viewing our algorithm as a message passing algorithm is advantageous becomes one can easily appreciate the distributed nature of our solution, as well as its relationships to some related works such as [16, 23]. For an overview on message passing algorithms for inference on graphical models see [77].

To see each update as a message passed between two nodes, let us take a look at the update Eq. 3.65. This update equation can be interpreted as a message passed from node μ_{k+1} to f_k, given by $m^{(i)}_{f_k \leftarrow \mu_{k+1}} = e^{\alpha_{k+1} \log Z_{\mu_{k+1}}(\alpha_{k+1})}$. This message resembles a standard message in an inference problem, except for an extra term in the exponent, i.e., α_{k+1}. By comparing with a Boltzmann distribution, one can see that the the role of the term α_{k+1} can be interpreted as scaling down the *temperature* of a Boltzmann distribution with energy $\log Z_{\mu_{k+1}}$ by α_{k+1}. Similarly, one can interpret Eq. 3.66, as a combination of two messages; one passed from dynamics to policy, defined as $m^{(i)}_{\mu_k \leftarrow f_k} = e^{\beta_k \log Z_{f_k}(\beta_k)}$ and one passed from the cost node to policy, given by $m^{(i)}_{\mu_k \leftarrow r_k} = e^{-c_k}$. All these messages are closely similar to standard messages in inference, except for frequent temperature modulation of messages. Therefore, it is useful to define a temperature operator,$\mathcal{T}_\alpha$, which scales the temperature by

[6]Or more strictly, each iteration, does increase the objective, i.e., either decreases it or keeps it the same.

α. Similarly, the inverse operator $\mathcal{T}_\alpha^{-1}$, scales the temperature by α^{-1}. Therefore,

$$\mathcal{T}_\alpha\left[Z\left(\cdot\right)\right] = Z\left(\cdot\right)^{\frac{1}{\alpha}}, \mathcal{T}_\alpha^{-1}\left[Z\left(\cdot\right)\right] = Z\left(\cdot\right)^{\alpha}. \tag{3.69}$$

With the definition of this operator the messages become:

$$\boxed{m_{f_k\leftarrow\mu_{k+1}}^{(i)} = \mathcal{T}_{\alpha_{k+1}}^{-1}\left[\int_{\mathbf{u}_{k+1}} \frac{\mu_k^{(i)}\mathcal{T}_{\alpha_{k+1}}\left[m_{\mu_{k+1}\leftarrow f_{k+1}}^{(i)}\right]\mathcal{T}_{\alpha_{k+1}}\left[m_{\mu_{k+1}\leftarrow r_{k+1}}^{(i)}\right]}{\mathcal{T}_{\alpha_{k+1}}\left[m_{f_k\leftarrow\mu_{k+1}}^{(i)}\right]} d\mathbf{u}_{k+1}\right]} \tag{3.70}$$

$$\boxed{m_{\mu_k\leftarrow f_k}^{(i)} = \mathcal{T}_{\beta_k}^{-1}\left[\int_{\mathbf{x}_{k+1}} \frac{f_k\mathcal{T}_{\beta_k}\left[m_{f_k\leftarrow\mu_{k+1}}^{(i)}\right]}{\mathcal{T}_{\beta_k}\left[m_{\mu_k\leftarrow f_k}^{(i)}\right]} d\mathbf{x}_{k+1}\right]} \tag{3.71}$$

$$\boxed{m_{\mu_k\leftarrow r_k}^{(i)} = \mathcal{T}_{\eta_k}^{-1}\left[\sum_{r_k=0}^{1} p\left(r_k\right)\right]} \tag{3.72}$$

The update equations in message passing form become,

$$\boxed{f_k^{(i+1)} = \frac{f_k\mathcal{T}_{\beta_k}\left[m_{f_k\leftarrow\mu_{k+1}}^{(i)}\right]}{\mathcal{T}_{\beta_k}\left[m_{\mu_k\leftarrow f_k}^{(i)}\right]}} \tag{3.73}$$

$$\boxed{\mu_k^{(i+1)} = \frac{\mu_k^{(i)}\mathcal{T}_{\alpha_k}\left[m_{\mu_k\leftarrow f_k}^{(i)}\right]\mathcal{T}_{\alpha_k}\left[m_{\mu_k\leftarrow r_k}^{(i)}\right]}{\mathcal{T}_{\alpha_k}\left[m_{f_{k-1}\leftarrow\mu_k}^{(i)}\right]}} \tag{3.74}$$

One can see that the scale of the temperature of the messages entering a node is changed to that of the node, and when exiting a node, it is normalized. Accordingly, one can view each of μ_k, f_k, and r_k to be in an environment with their own temperature unit. When messages go from one environment to the other, their temperatures has to be scaled. Fig. 3.7 is a depiction of these messages and explains how the temperature should be modulated from one node to the other.

Also, one can easily verify that if all the temperatures are taken to be the same, the messages reduce to those of a message passing solution of the posterior of the dynamics and the policies, given the observations, $\Pr\left(R_k = 1 \mid \mathbf{X}_k = \mathbf{x}_k, \mathbf{U}_k = \mathbf{u}_k\right) = e^{-\frac{c_k(\mathbf{x}_k,\mathbf{u}_k)}{\eta}}$. This special case is discussed in the following section.

3.6.4 The Special Case of Posterior Policy Iteration

Let's assume, $\alpha_k = \beta_k = \eta > 0$. In this case, the PP iterations in Prob. 10 reduce to,

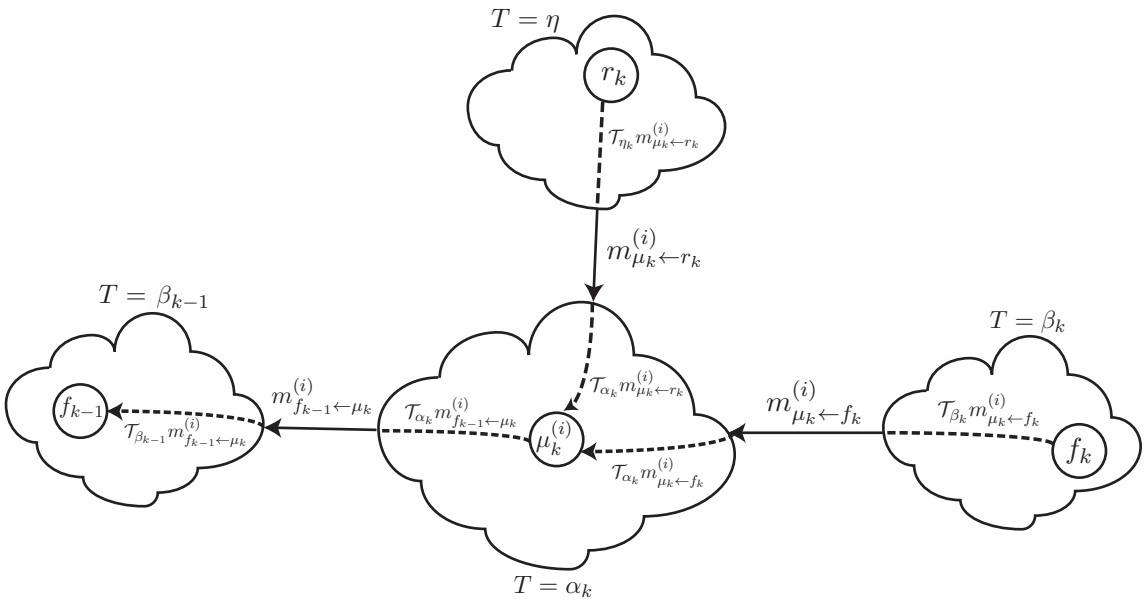


Figure 3.7 – Out message passing solution for SOC. The temperature of messages passed from one environment to the other has to be modulated.

$$\min_{p'_0, \mu'_k, f'_k} \eta \int_{d\bar{\mathbf{x}} d\bar{\mathbf{u}}} p'_0 \prod_{k=0}^{N} \mu'_k f'_k \log \frac{p'_0 \prod_{k=0}^{N} \mu'_k f'_k}{p_0^{(i)} \prod_{k=0}^{N} \mu_k^{(i)} f_k^{(i)} e^{-\frac{c_{0:N}}{\eta}}} d\bar{\mathbf{x}} d\bar{\mathbf{u}} \tag{3.75}$$

$$= \min_{q'} \eta \mathrm{KL}\left(q' \parallel q^{(i)} e^{-\frac{c_{0:N}}{\eta}}\right). \tag{3.76}$$

The minimization above can be solved as a standard posterior inference, which due to the factorized structure of q can be solved efficiently using message passing algorithms [77]. The terms, $e^{-\frac{c(\bar{\mathbf{x}}, \bar{\mathbf{u}})}{\eta}} = e^{-\frac{c_0}{\eta}} e^{-\frac{c_1}{\eta}} \cdots e^{\frac{c_N}{\eta}}$ can be considered as likelihoods of some auxiliary observations. Therefore, one can define auxiliary binary random variables, $R_0 \cdots R_N$, ('R' as in reward)[7] and define their probability of being observed to be $'1'$, as

$$\Pr\left(R_k = 1 \mid \mathbf{X}_k = \mathbf{x}_k, \mathbf{U}_k = \mathbf{u}_k\right) = e^{-\frac{c_k(\mathbf{x}_k, \mathbf{u}_k)}{\eta}}. \tag{3.77}$$

Therefore, one can represent this problem using a graphical model. A Bayesian network representation is given in Fig. 3.8. This solution is known as *posterior policy iteration* (PPI) as described in [16]. However, here it was derived as a special case where all multipliers are chosen equally.

[7] It should be noted that these auxiliary variables can be defined arbitrarily, as long as they are observed to some event with likelihood proportional to $e^{-\frac{c_k(\mathbf{x}_k, \mathbf{u}_k)}{\eta}}$. Here we choose R, as in reward, because the likelihood of R decreases as the cost increases and, therefore, such naming seems very appropriate.

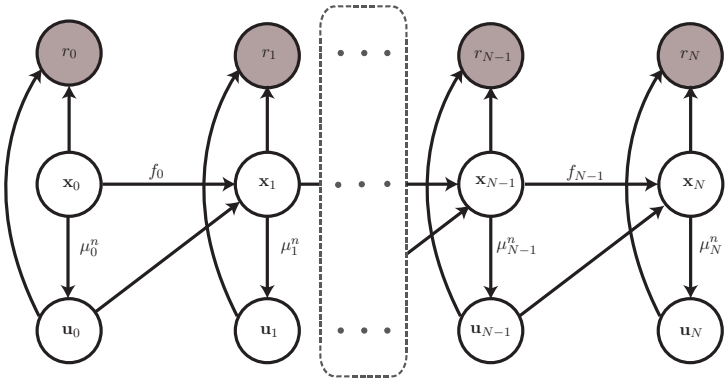


Figure 3.8 – Bayesian network representation of Posterior Policy Iteration

The message passing solution to the posterior inference on the Bayesian net in Fig. 3.8 bears some similarities to our backward updates. The difference lies in the fact that with our backward updates, the conditional distributions are inferred directly, whereas with standard message passing, one has to infer the posterior marginals as a first step, and then derive the posterior conditionals by marginalization. Because of this difference, the messages passed on the Bayesian chain consist of a forward pass and a backward pass, while in our case, there was no forward flow of information and we only had the backward updates of Sec. 3.6.1. Indeed, to infer the posterior conditionals one does not need any forward message as they will be integrated out.

Finally, PPI is known to be risk-seeking [16]. The risk-seeking property of PPI can easily be explained by noting that positive and finite choice of the multipliers ,β_k, results in an optimistic and risk-seeking property of the policy. This risk-seeking effect can also be seen from the perspective of risk sensitive control and by realizing that the posterior policy minimizes $\left\langle e^{-\frac{c}{\eta}} \right\rangle_{q_\pi}$, which is exactly the risk sensitive objective as defined in [68] . It can be shown that for large values of $|\eta|$,

$$\eta \log \left\langle e^{-\frac{c}{\eta}} \right\rangle_{q_\pi} = \langle c \rangle_{q_\pi} - \frac{1}{\eta} \mathrm{cov}_{q_\pi}(c) + \text{higher order terms.} \tag{3.78}$$

This means that for positive values of η, minimizing $\left\langle e^{-\frac{c}{\eta}} \right\rangle_{q_\pi}$, gives policies with increased cost covariance. This interpretation is also mentioned in [16]. However, our own interpretation of η in connection with the radii of KL-ball constraints for the policies and dynamics is more general and can explain more general cases where α_k and β_k are chosen differently.

3.7 An Extended Kalman Filter Approximation

In Sec. 3.4.5, it was shown that each KL-penalized problem can be interpreted and solved as a standard posterior problem. Therefore, any approximate inference method can be applied to solve each of the updates in Eqs. 3.65 –3.68. Some of the approximations that can be

used include, approximations based on extended Kalman filtering (EKF) [97], unscented Kalman filtering, [98], particle methods [99], expectation propagation [100] and MCMC methods [19]. Here, results based on EKF-type approximations are considered, because of the simplicity of their implementation, and also, because despite it's simplicity, EKF is quite effective in dealing with nonlinear filtering problems. Naturally, just like the filtering problem, using more accurate approximate methods, results in increased accuracy and globalist of the solution, at the expense of increased computational load. Therefore, applying more advanced inference methods to approximate Eqs. 3.65 –3.68 is a worthy direction for future research.

In the following, an algorithm for computing update Eqs. 3.65–3.68 based on EKF-type approximations is described. The notation used for denoting Gaussian distributions, along with other related definitions are detailed in Appx. B. Also, fore a review on Kalman filtering and EKF see [97] and also Appx. D.

With EKF-type approximations, one assumes that the stochasticity in the dynamics is Gaussian (or that it can be well approximate by one). That means that the *nonlinear* dynamics has the following form,

$$f_k\left(\mathbf{x}_{k+1} \mid \mathbf{x}_k, \mathbf{u}_k\right) = \mathcal{N}\left(\mathbf{x}_{k+1} \mid \mathbf{f}_k\left(\mathbf{x}_k, \mathbf{u}_k\right), \Sigma_{\mathbf{w}}\right), \tag{3.79}$$

where $\Sigma_{\mathbf{w}}$ is the covariance matrix for the additive Gaussian process noise. In order to apply EKF, the first step is to linearize the above nonlinear dynamics around an initial state-control trajectory[8] $\bar{\mathbf{x}}^{(0)}, \bar{\mathbf{u}}^{(0)}$. That is,

$$f_k\left(\mathbf{x}_{k+1} \mid \mathbf{x}_k, \mathbf{u}_k\right) \simeq \mathcal{N}\left(\mathbf{x}_{k+1} \mid \mathbf{A}_k\mathbf{x}_k + \mathbf{B}_k\mathbf{u}_k + \mathbf{a}_k, \Sigma_{\mathbf{w}}\right). \tag{3.80}$$

where,

$$\mathbf{A}_k = \left.\frac{\partial \mathbf{f}_k\left(\mathbf{x}_k, \mathbf{u}_k\right)}{\partial \mathbf{x}_k}\right|_{\mathbf{x}_k=\mathbf{x}_k^{(0)}, \mathbf{u}_k=\mathbf{u}_k^{(0)}}, \tag{3.81}$$

$$\mathbf{B}_k = \left.\frac{\partial \mathbf{f}_k\left(\mathbf{x}_k, \mathbf{u}_k\right)}{\partial \mathbf{u}_k}\right|_{\mathbf{x}_k=\mathbf{x}_k^{(0)}, \mathbf{u}_k=\mathbf{u}_k^{(0)}}, \tag{3.82}$$

$$\mathbf{a}_k = \mathbf{f}_k\left(\hat{\mathbf{x}}_k, \hat{\mathbf{u}}_k\right) - \left(\mathbf{A}_k\hat{\mathbf{x}}_k + \mathbf{B}_k\hat{\mathbf{u}}_k\right). \tag{3.83}$$

The (possibly non-quadratic) cost function is quadratized around the same state-control trajectory, $\bar{\mathbf{x}}^{(0)}, \bar{\mathbf{u}}^{(0)}$, and approximated by the following quadratic form,

$$c_k\left(\mathbf{x}_k, \mathbf{u}_k\right) \simeq \frac{1}{2}\mathbf{x}_k^T\mathbf{Q}_k\mathbf{x}_k + \frac{1}{2}\mathbf{u}_k^T\mathbf{R}_k\mathbf{u}_k + \mathbf{x}_k^T\mathbf{C}_{\mathbf{xu},k}\mathbf{u}_k - \left(\mathbf{x}_k^T\check{\mathbf{x}}_k^* + \mathbf{u}_k^T\check{\mathbf{u}}_k^*\right) + const. \tag{3.84}$$

[8]The initialization doesn't have to be very informative. Our algorithm works for non-informative and trivial initializations. However, if a high-level path planning algorithm is available for the problem, one may use it for initialization.

Since the approximation of the cost function is quadratic, the auxiliary reward variables will have a Gaussian form[9]:

$$\Pr\left(R_k = 1 \mid \mathbf{X}_k = \mathbf{x}_k, \mathbf{U}_k = \mathbf{u}_k\right) \propto \mathcal{N}\left[\begin{bmatrix} \mathbf{x}_k \\ \mathbf{u}_k \end{bmatrix} \mid \eta^{-1}\begin{bmatrix} \check{\mathbf{x}}_k^\star \\ \check{\mathbf{u}}_k^\star \end{bmatrix}, \eta^{-1}\begin{bmatrix} \mathbf{Q}_k & \mathbf{C}_{\mathbf{xu},k} \\ \mathbf{C}_{\mathbf{xu},k}^T & \mathbf{R}_k \end{bmatrix}\right]. \tag{3.85}$$

Also, it is assumed that the initial guess[10] for the policy has the following form,

$$\mu_k^{(0)} = \mathcal{N}\left(\mathbf{u}_k \mid \mathbf{K}_k^{(0)}\mathbf{x}_k + \mathbf{u}_k^{ol(0)}, \Sigma_{\mu_k^{(0)}}\right), \tag{3.86}$$

where $\mathbf{K}_k^{(0)}$ is an initial control gain, $\mathbf{u}_k^{ol(0)}$ is an initial open-loop control signal and $\Sigma_{\mu_k^{(0)}}$ is the covariance matrix for the initial policy. This policy covariance matrix describes the uncertainty in the control policy; the larger the covariance matrix[11] , the more the uncertainty in the policy.

The update Eqs. 3.66–3.67 can now be performed using standard posterior calculation for each update, which will be the same as extended Kalman smoothing along the path $\bar{\mathbf{x}}^{(0)}, \bar{\mathbf{u}}^{(0)}$. The complete derivation of Gaussian update equations is given in Appx. C and it is shown that with an initial Gaussian policy, all updated policies remain Gaussian as well.

The Gaussian updates described in Appx. C can be applied to the problem, starting from the last time slice and going backward in time (synchronous) or updating each time slice in any desired order (asynchronous). Specifically, the update Eqs. C.25–C.30 are used for computing messages passed from dynamics to policies, and Eqs. C.48–C.55 for updating policies.

After finding the updated values for $\mathbf{K}_k^{(1)}$, $\mathbf{u}_k^{ol(1)}$, $\Sigma_{\mu_k^{(1)}}$, the initial trajectory, $\bar{\mathbf{x}}^{(0)}, \bar{\mathbf{u}}^{(0)}$, is revised by applying the updated policy to the system at initial state $p_0\left(\mathbf{x}_0\right)$. This is done by iteratively drawing a random control signal from the distribution,

$$\mathbf{u}_k^{(1)} \xleftarrow{rand.} \mu_k^{(1)}\left(\mathbf{u}_k \mid \mathbf{x}_k^{(1)}\right), \tag{3.87}$$

and applying it to the dynamics. Then a sample is drawn from the resulting distribution,

$$\mathbf{x}_{k+1}^{(1)} \xleftarrow{rand.} f_k\left(\mathbf{x}_{k+1} \mid \mathbf{x}_k^{(1)}, \mathbf{u}_k^{(1)}\right). \tag{3.88}$$

[9]The likelihood of the auxiliary variable is formed by exponentiating the cost function. The exponential of a quadratic form is a multivariate Gaussian

[10]Again, the initial guess can be non-informative or informative. An initial zero gain and zero open-loop control worked just fine in all cases during our evaluations.

[11]Larger, as in the positive definite matrix sense. Indeed, the covariance matrix is positive definite, and positive definite matrices can be ordered. For instance, if the algorithm works, it is expected that $\Sigma_{\mu_k^{(i)}} \succeq \Sigma_{\mu_k^{(0)}}$.

Algorithm 3.1 EKF-type approximation of general PP update Eqs. 3.66–3.67

Input:

Dynamics: $f_k\left(\mathbf{x}_{k+1} \mid \mathbf{x}_k, \mathbf{u}_k\right)$

Initial State Distribution: $p_0\left(\mathbf{x}_0\right)$

KL-Penalty Coefficients: $\alpha_k, \beta_k \mid k \in \{0 \cdots N\}$

Initial guess for policies. Default: $\mathbf{K}_k^{(0)} = \mathbf{0}_{m \times n}, \mathbf{u}_k^{ol(0)} = \mathbf{0}_{m \times 1}, \Sigma_{\mu_k^{(0)}} = \mathbf{I}_{m \times m}$.

1: i=0

2: **while** !(Maximum no. of iterations reached) **or** !(Converged) **do**

3: $\mathbf{x}_0^{(i)} \longleftarrow p_0\left(\mathbf{x}_0\right)$. # Forward stochastic simulation.

4: **for all** $k \in \{0 \cdots N\}$. **do**

5: $\mathbf{u}_k^{(i)} \xleftarrow{rand.} \mu_k^{(i)}\left(\mathbf{u}_k \mid \mathbf{x}_k^{(i)}\right)$.

6: $\mathbf{x}_{k+1}^{(i)} \xleftarrow{rand.} f_k\left(\mathbf{x}_{k+1} \mid \mathbf{x}_k^{(i)}, \mathbf{u}_k^{(i)}\right)$.

7: $\bar{\mathbf{x}}^{(i)} = \left\{\mathbf{x}_0^{(i)}, \cdots, \mathbf{x}_N^{(i)}\right\}, \bar{\mathbf{u}}^{(i)} = \left\{\mathbf{u}_0^{(i)}, \cdots, \mathbf{u}_N^{(i)}\right\}$.

8: $\{\mathbf{A}_k, \mathbf{B}_k, \mathbf{a}_k \mid k \in \{0 \cdots N\}\} \leftarrow$Linearize dynamics around $\bar{\mathbf{x}}^{(i)}, \bar{\mathbf{u}}^{(i)}$.

9: $\{\mathbf{Q}_k, \mathbf{R}_k, \mathbf{C}_{\mathbf{xu},k}, \check{\mathbf{x}}_k^\star, \check{\mathbf{u}}_k^\star \mid k \in \{0 \cdots N\}\} \leftarrow$Quadratize cost around $\bar{\mathbf{x}}^{(i)}, \bar{\mathbf{u}}^{(i)}$.

10: Choose a desirable update *schedule* from any permutation of $\{0 \cdots N\}$. Default:$\{N, N-1, \cdots, 0\}$

11: **for all** $k \in$ *schedule* **do**

12: $\left\{\mathbf{K}_k^{(i+1)}, \mathbf{u}_k^{ol(i+1)}, \Sigma_{\mu_k^{(i+1)}} \mid k \in \{0 \cdots N\}\right\} \leftarrow$Update policies using Eqs. C.25–C.30 and C.48–C.55

13: i++

Output: The policy after i iterations: $\mathbf{K}_k^{(i+1)}, \mathbf{u}_k^{ol(i+1)}, \Sigma_{\mu_k^{(i+1)}}$

Performing the above from $k = 0$ to $k = N$ is referred to as a forward stochastic simulation, and it results in a first revision of the state-control trajectory $\bar{\mathbf{x}}^{(1)}, \bar{\mathbf{u}}^{(1)}$. The dynamics and the cost are then linearized and quadratized around this new trajectory $\bar{\mathbf{x}}^{(1)}, \bar{\mathbf{u}}^{(1)}$, and the update equations are applied to update the policy. This iteration is detailed in Algorithm 3.1, and continues until convergence is reached.

3.7.1 Choice of α_k and β_k

The choice of the set of parameters, α_k, and β_k defines the behavior of the resulting policy, as well as convergence rate of the algorithm. The proximal penalty coefficients,α_k, penalizes the new policy for getting too far from the old one. Therefore, increasing α_k, slows down the convergence for the decision, $\mu_k^{(i)}$. For β_k, from the discussion in Sec. it is known that for positive values of β_k, the optimal distribution is a minimizing solution, whereas for negative values of β_k , the optimal distribution is a maximizing solution. Therefore, for positive values of β_k, one has Prob. , which is the risk-seeking problem, whereas for negative values of β_k, one has Prob. ,which is the robust problem. Finally, for $\beta_k \to \infty$ one recovers the the standard SOC. Also, from the discussion in Sec. it's known that the inverse of$|\beta_k|$ has a direct

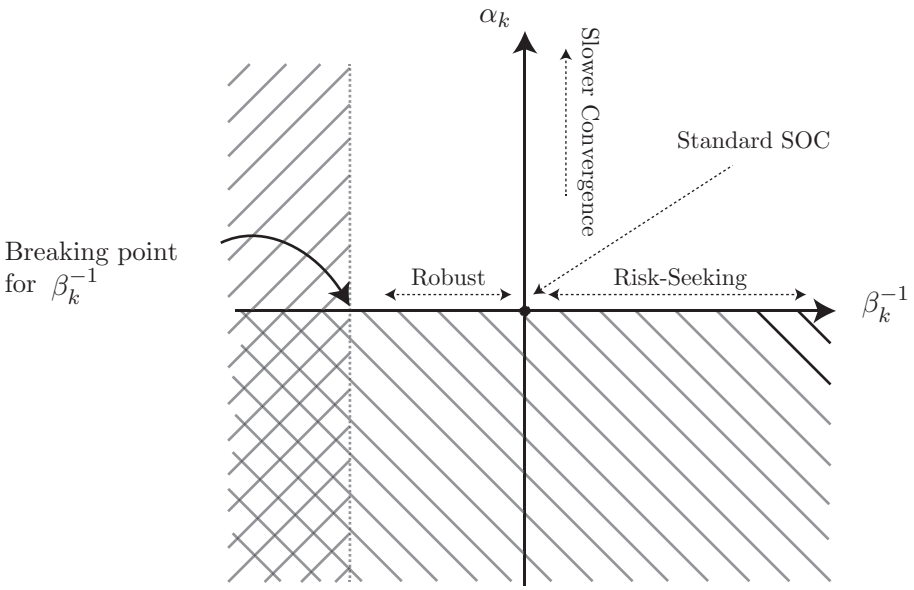


Figure 3.9 – The effect of choice of α_k andβ_k, on the convergence rate and the behavior of resulting policy.

relationship with the radius of the ambiguity set. This means that choosing a large value for $\left|\beta_k^{-1}\right|$ increases the the robust, or risk-seeking behavior. Moreover, for some negative values of β_k, the partition function in Eq. may not be well-defined. This corresponds to the case where the ambiguity set is so large, that there exists a worst-case dynamics under which, the maximum of the expected cost is infinity. That is, the robust problem is too pessimistic.

The effect of these parameters on the convergence rate and behavior of the policy is summarize in Fig. 3.9.

3.7.2 Application to the Cart-Pole Problem

In order to compare the presented algorithm, the nonlinear cart-pole problem is chosen. The cart-pole problem is often used as a test-bench for evaluating and comparing the performance of nonlinear control algorithms. Fig. 3.10. The objective of control is to balance the pole at $\theta = 0$, while positioning the cart at the origin $x_{cart} = 0$. The state vector is defined as,

$$\mathbf{x}_{cp} = \begin{bmatrix} x_{cart} & \dot{x}_{cart} & \theta & \dot{\theta} \end{bmatrix}^T. \tag{3.89}$$

The dynamics of the cart pole is governed by,

$$\frac{d}{dt}\begin{bmatrix} \theta \\ \dot{\theta} \\ x_{cart} \\ \dot{x}_{cart} \end{bmatrix} = \begin{bmatrix} \dot{\theta} \\ \frac{g\sin(\theta(m_c+m_p)) - \left(F+m_c l\dot{\theta}^2\sin\theta\right)\cos\theta}{\frac{4}{3}l(m_c+m_p) - m_p l\cos^2\theta} \\ \dot{x}_{cart} \\ \frac{F - m_p l\left(\ddot{\theta}\cos\theta - \dot{\theta}^2\sin\theta\right)}{m_c+m_p} \end{bmatrix}, \tag{3.90}$$

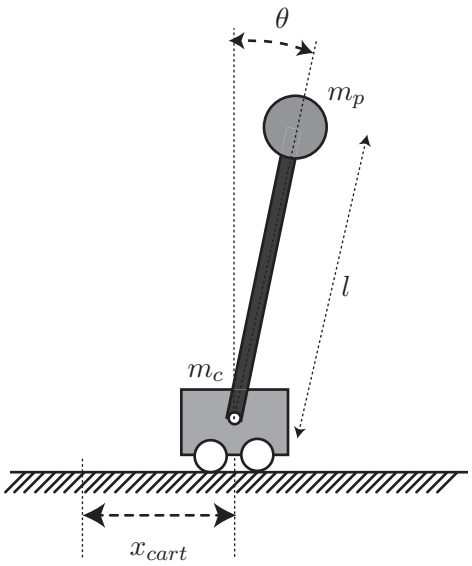


Figure 3.10 – The cart-pole problem. The objective of the control is to balance the pole, as well as positioning the cart on the origin.

Parameter	Symbol	Nominal value
Pole mass	m_p	0.5 Kg
Cart mass	m_c	1 Kg
Pole Length	l	1 m

Table 3.1 – Parameters of the cart-pole problem.

The parameters of the above model are defined in Table 3.1, and the control input is the force applied to the cart, F. The dynamics of cart-pole above is simulated using a fourth-order Runga-Kutta method, and to create stochasticity, a zero mean Gaussian noise with small covariance ($1e-5 \times diag\left([0, 5, 1, 1, 1]\right)$) is added to the predictions of the deterministic model above. The cost function is defined as the quadratic form below,

$$c(\mathbf{x}_{cp}) = \mathbf{x}_{cp}^T diag\left(\begin{bmatrix}0.1 & 0.01, & 1 & 0.01\end{bmatrix}\right)\mathbf{x}_{cp} + 0.01F^2. \tag{3.91}$$

The above cost function is minimized when the pole is balanced and the cart is positioned correctly, while giving the balancing task more priority by assigning it larger weight. To compare our method with standard SOC, the iLQG method [61] was also implemented for the problem. When iLQG converges *correctly* it finds a locally valid optimal policy to the problem. iLQG is chosen because it's been shown to have very fast convergence rate and also because it's been used for comparing SOC algorithms [16, 27, 28]. Specifically, the goal is to compare the convergence rate and convergence success rate and to showcase our robust SOC methodology compare to the standard SOC policy.

3.7.2.1 Convergence Rate and Convergence Success

First, the convergence rates for the two algorithms are compared. iLQG has second order convergence rate, and so, it's already quite fast. However, our algorithm also enjoys second order convergence rate, in addition to some advantages which further increase its convergence rate in practice. Specifically, our algorithm uses a stochastic forward pass, as opposed to the deterministic procedure in iLQG. The randomness in the forward pass allows it to escape local minima and steep regions, while iLQG spends many iterations near steep regions. This is similar to stochastic optimization methods such as simulated annealing, where the added stochasticity allows the algorithm to escape local minima.

Moreover, iLQG needs complicated regularization and/or a line search method to achieve successfully converge. For our comparison, an advanced implementation of iLQG which utilizes a backtracking line search in the inner loops was used. This extra step, adds to the number of forward passes, but ensures a greedy decrease of expected cost in each iteration.

Our method, however, enjoys a built-in regularization through the the set of α_k parameters, which control the proximity penalty. That is, they control how far it is allowed for the updated policy to go from the current policy. For our simulations, it was found that $\alpha_k = 1E-8$ was good enough to ensure successful convergence for all cases. This built-in regularization was also found to be fairly insensitive to the choice of, α_k, and changes within 2 order of magnitude did not effect convergence rate or convergence success noticeably. However, for larger choice of α_k, the regularization starts to be conservative and, so, the convergence rate begins to slow down. However, for some problems with sharp cost functions or other irregularities, increasing α_k trades convergence rate for convergence success. Fig. 3.11 shows the convergence rate of our approach for a fixed value of $\alpha_k = 1E-8$ and different choices of β_k, in comparison to iLQG. It is observed that the convergence rate speed is improved. This fast speed was also achieved by other inference based methods such as the work reported in [28], although our work enjoys more control over convergence through independent choice of α_k for each time step, k.

However, the true power of our approach comes from the fact that it can reproduce robust and risk-seeking control, without any increase in the computational load.

3.7.2.2 Robust and Risk-Seeking Control

As seen earlier, our algorithm is able to produce robust and risk-seeking through the choice of β_k. It was shown that for a negative choice of β_k robust control is resulted, while for positive choices, one reaches risk-seeking policies. In this section, the behavior of the optimal policy produced by our algorithm, for several choices of β_k is evaluated, compared to iLQG. It should be noted that for $\beta_k \to \infty$ our algorithm results in standard SOC, and therefore, it should result in the same average cost and variance as iLQG.

To compare the policies with different behaviors, several policies have been derived using our algorithm. Then, the policies were applied to a 1000 simulations of cart-pole system

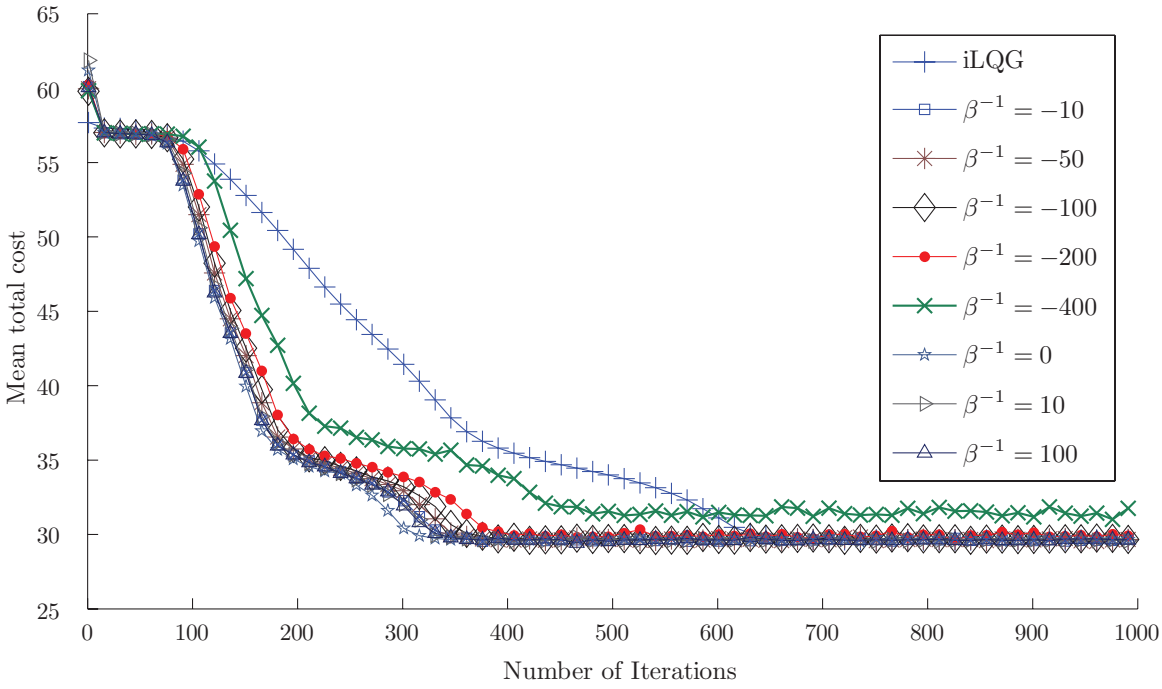


Figure 3.11 – Convergence rate comparison of iLQG vs. our method for $\alpha = 10^{-8}$, and different choices of β. In most cases, our algorithm convergence within 400 iterations, whereas it takes iLQG about 650 iterations to converge.

with 1%, 3% and 5% perturbation in the parameters of the system. The the cost resulted from iLQG and policies with different robustness (or risk-seeking) behavior were recorded for 1000 trials, and the average and the variance of the total cost was compared. The results are presented in Fig. 3.12. It's observed that when β_k is chosen as negative values, the cost variance is considerably lower than the cost variance resulted from applying iLQG. Also, for positive β_k the risk-seeking control results in increased cost variance. These results match our expectations based on the theory developed in this chapter, and show that with the same, or lower computational burden, it is able to derive robust control laws which achieve the same mean expected cost in low-uncertainty cases, while achieving much lower cost variance for cases when the uncertainty in the parameters is high. Fig. 3.12 show that the cost variance for our robust control policy can be as much as 5 times lower than the standard stochastic optimal policy calculated using iLQG. It is also shown that for the risk-seeking case, the cost variance is actually increased. The risk-seeking case was already derived through an inference based method in [16], while the results presented here show that all the three cases of risk-seeking, standard, and robust control can be reproduced by sweeping a parameter.

It's also useful to compare the time varying control gain matrix, resulted from robust control and risk-seeking control vs. the standard SOC. Fig. 3.13. A first inspection suggests that in all cases, the robust gain is lower while the risk-seeking gain is larger (in absolute value) than the standard SOC gain. However, the control gain is not simply different by a scale factor, as any attempt to reproduce one control gain by simply scaling the other failed.

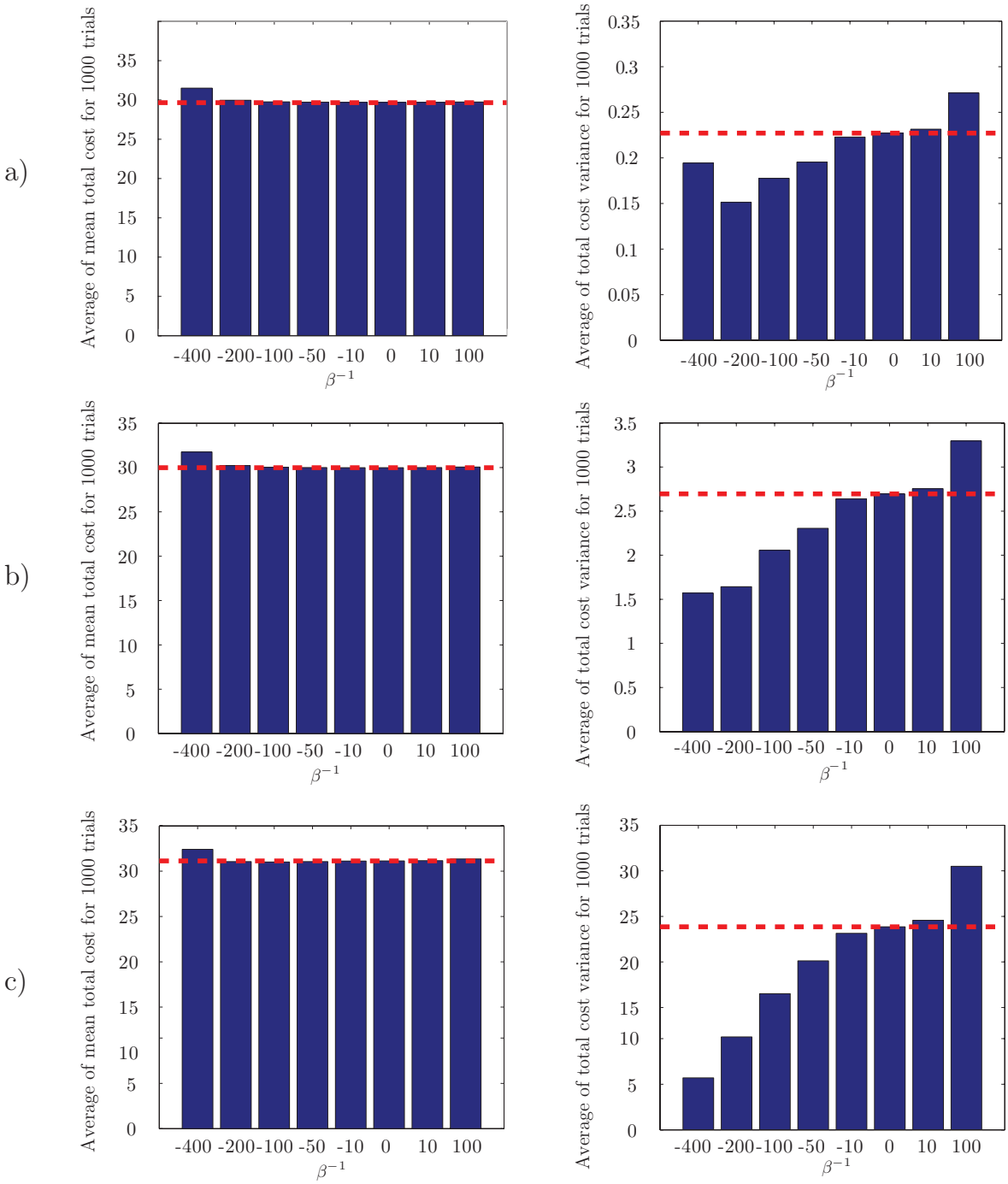


Figure 3.12 – Comparison of the average of mean total cost and total cost variance for 1000 trials of the cart-pole problem with a) 1% parameter perturbation b) 3% parameter perturbation c) 5% parameter perturbation. In all cases, the dashed (red) line corresponds to the benchmark results for iLQG.

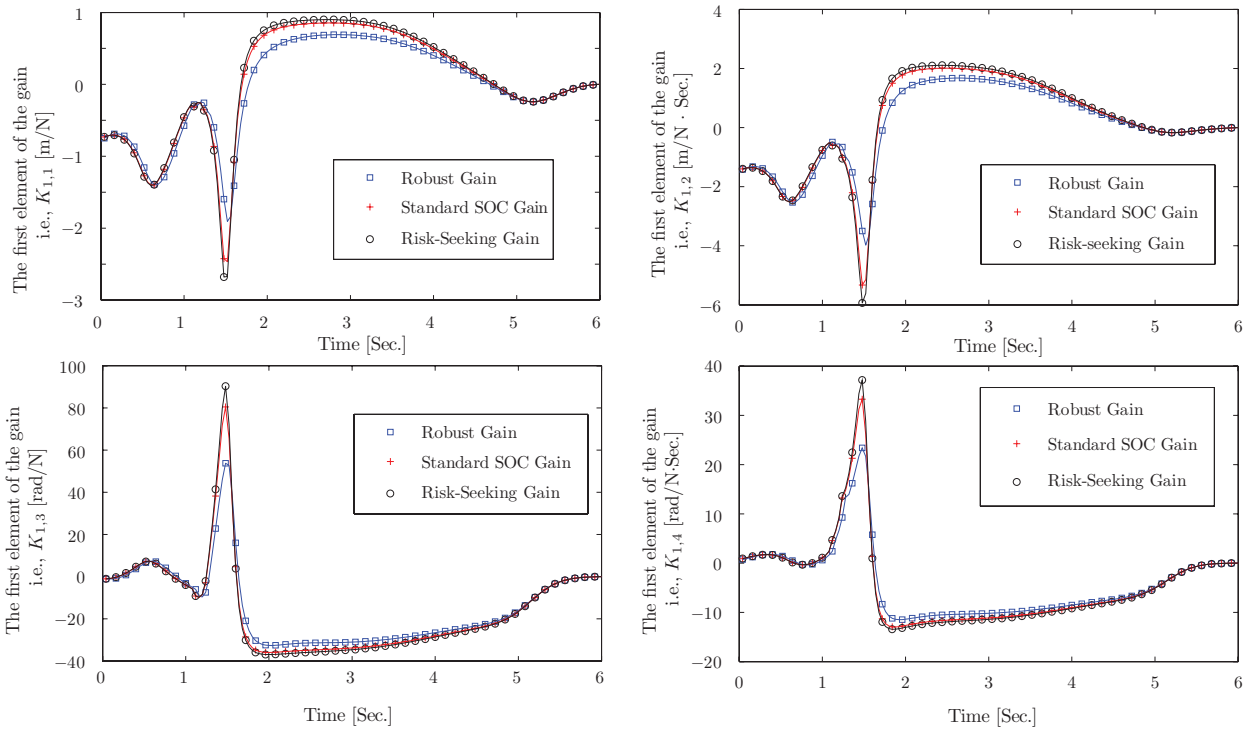


Figure 3.13 – Comparison of the gain matrix for a robust policy the standard and a risk-seeking policy.

3.7.3 Advantages of Our Solution

- Computational Efficiency: Compared to standard DP, the EKF approximation of our approach doesn't suffer from curse of dimensionality, by reducing the computation to a relevant region of the state and control space. This is the same technique as SDDP and iLQG, however, it was shown that our method outperforms iLQG because of superiority in dealing with local minima.

- Analytical tractability: The update Eqs. 3.65–3.68 lend themselves to analytical solutions for large classes of problems. Specifically, analytical solutions for LQ cases were derived in this section.

- Generality: Our solution can be applied to problems with any form of nonlinear dynamics, non-Gaussian noises and non-quadratic cost function. Therefore, it applies to the most general class of SOC problems.

- Flexibility: Our solution subsumes robust, optimistic and standard SOC, because each of these three variations can be derived as a special case of the solution to the PP problem above. For $\beta_k, \gamma_0 > 0$ one has optimistic control, for $\beta_k, \gamma_0 < 0$, robust control and for $\beta_k, \gamma_0 \to +\infty$, one recovers the standard SOC. The Lagrange multipliers, β_k, control the optimistic/pessimistic behavior of the policy *locally* in time. That is, the policy can be pessimistic at time $k = 4$, by a choice of $\beta_4 < 0$, while optimistic at time

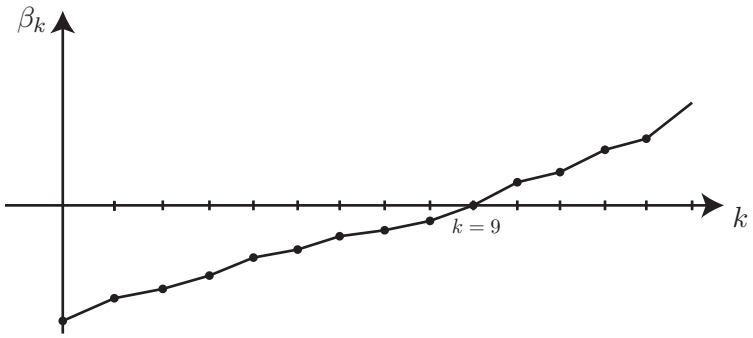


Figure 3.14 – A controller with varying robustness behavior. The controller starts with a pessimistic assumption about the available model of the plant and gradually shifts towards an optimistic behavior.

$k = 5$, by choosing$\beta_k > 0$. This allows the designer to define new types of controllers with a mixed pessimistic/optimistic behavior. This added flexibility has never been possible before. Fig. 3.14 depicts an example controller with a certain choice of β_k that results in an initially pessimistic (or robust) behavior and slowly shifts towards optimistic (or risk-seeking) behavior, as time passes. For instance, if the model of the system, f_k, is being learned or updated through a adaptive methodology, the designer can adjust the robustness based on how much trust he/she wants to put on the model.

3.8 Related Works

Our treatment of SOC can be related to a large set of works from different fields. However, for brevity only the most significant connections are discussed.

3.8.1 Linearly Solvable SOC

It was shown that problems with a weighted relative entropy term in the objective have a closed-form solution given by Lemma 1. A wide range of works in the literature take advantage of this fact. A large class of these works, referred to as linearly solvable SOC (or sometimes, *KL Control*) assume SOC problems with cost functions of the following special form:

$$c_k^{KL}\left(\mathbf{x}_k, \mathbf{u}_k\right) = c_k\left(\mathbf{x}\right) + \int_{\mathbf{u}_k} \mu_k\left(\mathbf{u}_k \mid \mathbf{x}_k\right) \log \frac{\mu_k\left(\mathbf{u}_k \mid \mathbf{x}_k\right)}{\mu_k^{passive}\left(\mathbf{u}_k \mid \mathbf{x}_k\right)} d\mathbf{u}_k, \tag{3.92}$$

for which closed-form solutions of the form in 3.28 can be derived. Also, it is shown that these problems can be linearized by an logarithmic transformation [22, 50], hence the title. Path integral control [101] is a special case of these problem. Also, it's been shown that these problems can be considered as a special case of Linearly Solvable Markov Games (LMGs)

[102]. All of these works assume the above form for cost function, and therefore, do not apply to general SOC problems, whereas the presented approach does.

3.8.2 Dynamic Policy Programming

A recent approach to SOC, namely the *dynamic policy programming* [56], and a later work [16] artificially inject a KL term into the cost function, in order to produce a sequence of problems with solutions converging to the optimal policy. This is, in fact, nothing but a PP method, which is used in these works without explicitly calling it so. The class of problems solvable by these approaches is more general than KL control problems, and includes very general nonlinear SOCs. However, our approach is more general than these works and reduces to these works as a special case.

3.8.3 Robust Control Approach of Sargent-Hansen

In Sec. 3.6 it was shown that for finite and negative values of the parameters,β_k and γ_0, the PP problem solves the robust control of [60, 92]. Also, each iteration of the PP algorithm, solves a robust control problem penalized by a proximity function on the policy. Therefore, our approach provides a novel solution to the robust control as a special case.

3.8.4 Rationality Bounded Games

In a seemingly disparate set of works, the concept of rationality bounded games are introduced, in which, a rationality cost (or information processing cost), expressed in terms of a (relative) entropy term is added to the cost function of agents in a two-player game. Using these ideas Ortega [57, 58] showed that a generalization to the Bellman equation can be derived. Also, they show how a two-player rationality bounded games can be reduced to SOC and robust control in the limits of rationality of the two agents.

In our work, *each iteration* of the PP problem can be interpreted as a rationality bounded game of the same form as in [47, 57]. In other word, rationality bounded games constitute the *inner loop* of our iterative algorithm. Thus, our approach can be considered as a sequence of rationality bounded games. The Ortega discusses how a penalty term of the form $\beta \mathrm{KL}\left(p\left(\mathbf{x}\right) \parallel p_0\left(\mathbf{x}\right)\right)$ can quantify the amount of (*computational*) *work* an agent has to do to transform an initial distribution, $p_0(\mathbf{x})$, to a solution which maximizes a utility, $\left\langle -c\left(\mathbf{x}\right)\right\rangle_{p(\mathbf{x})}$. The more computational resources available to the agent, the less the impact of computational cost on the result. Therefore, one can say that the weight of this penalty term is lower for more resourceful (or more rational) agents. A perfectly rational agent has unlimited computational resources and approach $p^{\star}\left(\mathbf{x}\right)$ as much as he/she wants, i.e. an agent with $\beta \to 0$.

Therefore, β^{-1}, can be interpreted as the *rationality parameter*. One can interpret *a single iteration* of our proximal point algorithm with Lagrange multipliers α_k, β_k and γ_0,

as a two player bounded rationality game with rationality parameters $\alpha_k^{-1}, \beta_k^{-1}$ and γ_0^{-1}. To see this interpretation more clearly, let's first identify the two players (or agents). In game theory, agents are epistemic metaphors which represent a (utility or cost) optimization problem. In the SOC problem, the controller tries to come up with a policy that minimizes the expected total cost. So the controller is clearly one of the agents. The plant, however, looks like a passive entity which merely follows a certain dynamics. Therefore, the plant in SOC is not a rational player. Fortunately, in rationality bounded game theory we can consider such an agent as an extreme case of a rationality bounded agent, i.e. , an agent with zero rationality, or, $\beta_k^{-1} \to 0$.

Each iteration of the optimistic and pessimistic proximal point solutions discussed in Sec. 3.5 can also be interpreted as a two player game with plant as the cooperative or adversarial second player, respectively. The adversarial case, which is when the plant tries to maximize the cost and the controller tries to minimize it, is very similar to a game-theoretic minimax problem where the perfectly rational players are replaced with rationality bounded ones. If the second player is cooperative, which is when, she/he, too, tries to minimize the total average cost, we have the optimistic.

It should be noted that while Ortega [57, 58] considers the case of fully rational adversarial plant, i.e., $\beta_k^{-1} \to -\infty$, as robust control, here we content that this extreme case is not interesting at all, because it assumes an arbitrary and worst case dynamics for the plant and solves the optimal policy for this dynamics. This worst case dynamics becomes increasingly irrelevant to the nominal dynamics of the plant as $\beta_k^{-1} \to -\infty$, and as a result, the policy also becomes increasingly over-conservative for the nominal dynamics and other plausible dynamics in its proximity. Also, it was shown that robust control approach of Sargent-Hansen calls for finite and reasonable radii for the reasonably chosen ambiguity set and it cannot be chosen to be arbitrarily large. Moreover, in Sec. 3.4.4 it was shown that the case $\beta_k^{-1} \to -\infty$ may not even be well defined.

3.8.5 Risk-Sensitive Control and Stochastic System with Relative Entropy Uncertainty Bounds

Stochastic system with relative entropy bounds on their uncertainty, i.e., the pessimistic case defined in Prob. 1, have been studied in [103], and their relationship to risk-sensitive control has been known. Specifically, [103] explains how a robust control problem with relative entropy bounds can be solved as a risk-sensitive control problem. The work in [103] also provides the solution for the LQG case. While the proposed method here uses the same problem formulation as in [103], the solution given here is applicable to problems with nonlinear dynamics, while the LQG results in [103] are only applicable to linear problems with quadratic costs. In fact, the proposed solution in this chapter is based on an iterative application of posterior calculation, and is inherently different from the method proposed in [103].

3.8.6 Other Related Works

There are several other inference-based works which have connections to our work. In some works, [17, 24], Expectation-Maximization (EM) is used to derive an iterative solution to SOC. Indeed, EM, *alternates* between finding the expected value of the log-likelihood for a set of parameters (the E-step), and maximizing this expected log-likelihood with respect to some parameters (the M-step). Therefore, EM can be considered as an instance of alternating optimization approach [96]. Our approach also finds the policy which maximizes the energy (or expected log-likelihood) of the total graph. However, our approach does this differently, with synchronous or asynchronous updates that improve the policy at each update.

Another recent work [23] studies message-passing solutions to general influence diagrams based on a proximal point algorithm, with a penalty function very similar to ours, except our penalty function is more general and allows for choosing the penalty coefficients to be chosen independent of each other.

3.9 Filtering Problem and Integration with SOC

It was shown that that the solutions to optimal control problem can be derived by a special type of inference. Especially, in Sec. 3.15 it was shown that a risk-sensitive policy can be derived by using standard posterior inference. This special case has been previously derived in [16] and is known as PPI. Since both filtering and SOC can be solve using inference, in this section it will be briefly shown how PPI and the filtering problem can be combined into a single inference problem and solved simultaneously and efficiently using message passing algorithms.

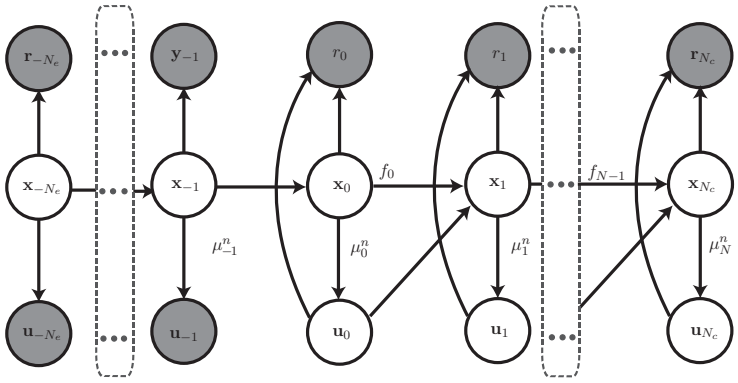


Figure 3.15 – PPI combined with filtering . More than one observation are used in order to get more accurate state estimation.

To integrate filtering and control, an integrated filtering/control graphical model can be created and solved for posterior marginals and subsequently, for policies. To solve the graph a variety of solutions to inference on graphical models can be used, including efficient message passing algorithms [77, 94]. Fig. 3.15 shows the combined filtering and optimal

control graphical model.

For a problem with quadratic costs and Gaussian noises, when a linearized model of the plant is used, the solution to the above is a standard extended Kalman smoothing, solve using a forward and backward sweep [104].

Chapter 4

Application to Visual Servoing

Visual servoing deals with the problem of controlling robotic systems through visual feedback. Research on visual servoing has been around for at least three decades, since the term was coined in 1979 [62]. Since then, numerous useful surveys and reviews have been written on the subject such as [105–109].

Visual servoing is a multifaceted field and relates to many disciplines such as *computer vision*, *control theory* and *robotics*. Visual information, i.e., images from cameras or other visual sensors, has to be processed to extract information relevant to control. This information is then fed to the controller which has to be designed in order to stabilize the complicated and nonlinear dynamics of the robot, in addition to controlling the robot according to a desired behavior. Fig. 4.1 illustrates a typical cycle and the various sub-problems faced in visual servoing. In this chapter, our focus is on the control problem and, therefore, vision problems and low-level control issues of the robot are assumed to be readily taken care off, using standard approaches. These are realistic assumptions that are assumed by most works in visual servoing [107].

The goal, in this chapter, is to apply our novel SOC solution to visual servoing and to implement it in real-time. Optimal control has several advantages over classical approaches

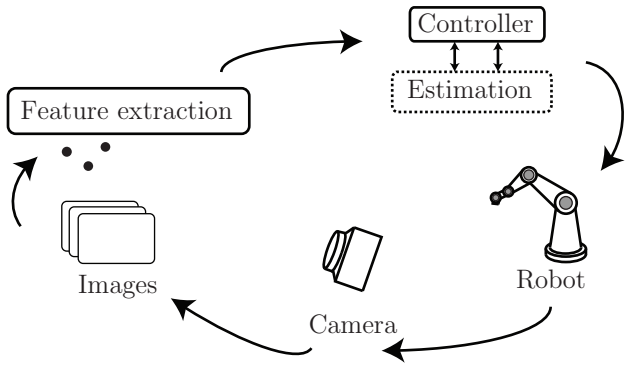


Figure 4.1 – A typical cycle in visual servoing demonstrating the diversity of the problem.

in visual servoing, and so, in recent years several works have tried to apply such advanced control approaches to visual servoing [110–113]. These works clearly show the improvements resulted from application of optimal control methods to visual servoing. In fact, it is natural to expect advanced control methods to show improvements over classical and much simpler controllers. The real challenge, however, is to implement these methods in *real-time.* This is not a trivial task, because such advanced control methodologies suffer from heavy computational loads, which don't simply allow a practical implementation for real-world applications. Consequently, most of the previous results on the use of advanced optimal-control-based methods to visual servoing were only verified through *simulations* [110, 111, 113].

Since one important aspect of the proposed SOC solution is its computational efficiency, it's a great candidate for *real-time* implementatio of optimal contro for visual servoing. In this chapter, the proposed SOC solution is tailored and modified for application to visual servoing, and experimental results of its successful real-time implementation are presented and compared with a standard visual servoing method to show its advantages.

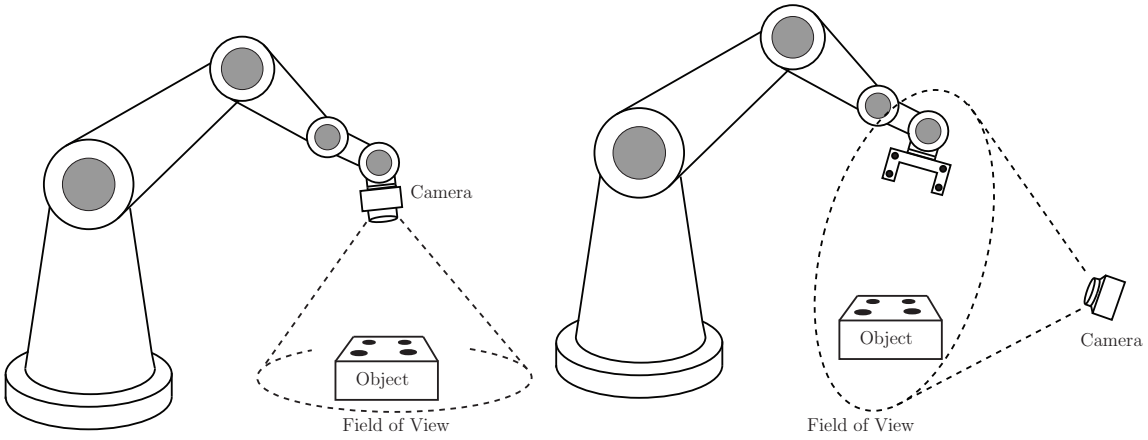
4.1 Preliminaries

There are many variations and flavors of visual servoing problems. First of all, the robotic system to be controlled can be a mobile robot, as well as an industrial robotic manipulator. In the case of industrial manipulators, which is the focus of this chapter, many configuration of camera (or cameras) w.r.t. the manipulator are possible. For a single camera, it can be mounted on top of the end-effector of the robot (i.e., eye-in-hand configuration) or placed statically in the work-cell of the robot to observe the robot and the environment together (i.e., eye-to-hand configuration) [107]. Of course, a combination of the two, as well as using stereo vision has also been studied. Fig. 4.2a depicts the eye-in-hand configuration and Fig. 4.2b illustrates the eye-to-hand configuration [105, 109].

Although due to its generality, given appropriate state-space modeling, the proposed control methodology can be applied to all of these problems, the goal of this chapter is to demonstrate its usefulness by applying it to a practically important case. Specifically, an eye-in-hand configuration with a single camera mounted on top of an industrial robot is assumed. The control objective of the visual servoing problem considered here is to drive the robot into a target pose (i.e. position and orientation) with respect to a target *object.* In order to represent these with more mathematical rigor, a brief review on kinematic notations follows.

4.1.1 Kinematic Notations

The kinematic notations used herein are similar to the tutorial on visual servoing in [109]. Accordingly, the position of a point in an arbitrary coordinate frame is denoted by ${}^{a}\mathcal{P}$, where the leading superscript, a, defines the coordinate frame. The superscripts, c, o, b, and



(a) The eye-in-hand configuration. **(b)** The eye-to-hand configuration.

Figure 4.2 – Two configurations of camera in visual servoing for industrial manipulators.

e are used to denote camera frame, object frame, base frame and the end-effector frame, respectively. The rotation matrix and the translation vector of frame b w.r.t frame a are denoted by ${}^a\mathcal{R}_b$ and ${}^a\mathcal{T}_b$, respectively. Also, ${}^a\mathcal{F}_b$ is used to denote the *pose* of frame b w.r.t. a, or equivalently, a frame transformation from b to a. Therefore,${}^a\mathcal{F}_b$ can be thought of as a standard homogenous transformation used very commonly in robotics [114]. The leading superscript specifying the frame may be dropped whenever it refers to the base (or the world) frame. Furthermore, the rotational, and translational velocity vectors are defined as $\Omega = [\omega_x, \omega_y, \omega_z]^T$and $\mathbf{V} = [v_x, v_y, v_z]^T$, respectively. These two can also be stacked to give the commonly known velocity screw $\boldsymbol{\nu} = [\mathbf{V}^T, \Omega^T]^T$. In order to transform a velocity screw from a coordinate frame to another, one can use,

$$ {}^b\boldsymbol{\nu} = \begin{bmatrix} {}^b\mathbf{V} \\ {}^b\Omega \end{bmatrix} = \begin{bmatrix} {}^b\mathcal{R}_a \, {}^a\mathbf{V} - {}^b\mathcal{R}_a \, {}^a\Omega \times {}^b\mathcal{T}_a \\ {}^b\mathcal{R}_a \, {}^a\Omega \end{bmatrix}. \tag{4.1} $$

Using 4.1 one can change the coordinates of the velocity screw of the camera from base frame to camera frame and vice versa. This relationship is helpful for conversion between frames since it is desirable to describe everything in the camera frame. Given the notations above, the objective of the control can now be described more rigorously. The goal of the control is to bring the pose of the camera w.r.t the object, ${}^o\mathcal{F}_c$, into a the desired pose, ${}^o\mathcal{F}_c^\star$, or equivalently, to bring the corresponding rotation matrix and translation matrix into desired ones, ${}^o\mathcal{T}_c^\star$ and ${}^o\mathcal{R}_c^\star$. In order to achieve this goal using visual feedback, the raw images from the camera are processed to extract the most relevant visual information, which are referred to as *visual features*.

4.1.2 Visual Features

For control purposes, one does not need to use all the information contained in raw images from camera. Typically, it is enough to base the control on a set of prominent visual features. For instance, corners, sharp edges, distinctive blubs or shapes such as circles and rectangles can be easily detected and tracked in a sequence of images, and used for matching and alignment, and ultimately, for controlling the pose of the robot. There exist more advanced visual features extraction mechanisms that can be used for visual servoing, such as scale-invariant features [115] used for visual servoing in [116]. However, these methods are more complex onto the image processing side, while the focus of this chapter is on the control methodology. Therefore, for our purposes, it suffices to consider two dimensional point features in the image, which can be detected from simple corner detection or by finding centers of high contrast blubs. Naturally, any image processing module which outputs a set of 2D points, (such as scale-invariant feature extraction procedures) can simply replace our basic blub detection module without modifying the controller.

Each point feature in the image can be described by two parameters, namely, u and v which are the horizontal and vertical pixel coordinate of the point, respectively. Each of these 2D image points is a projection of an actual 3D point, $\mathcal{P}$, on the camera sensor. The relationship between points in the environments and their projection on the image depends on the type of the camera and the types of lenses among other things. For instance, a fish-eye lens has a different mapping than a wide-angle lens. Even for typical lens-camera configurations the projection may not be a a simple perfect planar projection and there is always some image distortion. The mapping between actual 3D points and their 2D projection on the image is described by the *camera model*. The simplest camera model is the pinhole camera model, which is used in this chapter as well as nearly all works in visual servoing. In a pinhole model, the mapping is described by a perspective projection. Fig. 4.3 illustrates the projection of a 3D object onto the 2D *image plane* of the camera.

The relationships governing this projection are given by Eqs. 4.2–4.4.

$$
\begin{aligned}
{}^{c}\mathcal{P} &= \begin{bmatrix} X & Y & Z \end{bmatrix}^{T} && (4.2)\\
u &= \frac{X}{Z/f_c} + c_u && (4.3)\\
v &= \frac{Y}{Z/f_c} + c_u && (4.4)
\end{aligned}
$$

The parameters, f_c, c_v and c_u are the focal length and the vertical and horizontal center pixel coordinates, respectively. These parameters can be derived from a standard camera calibration procedure, such as the one described in [117]. It's also useful to define the following quantities,

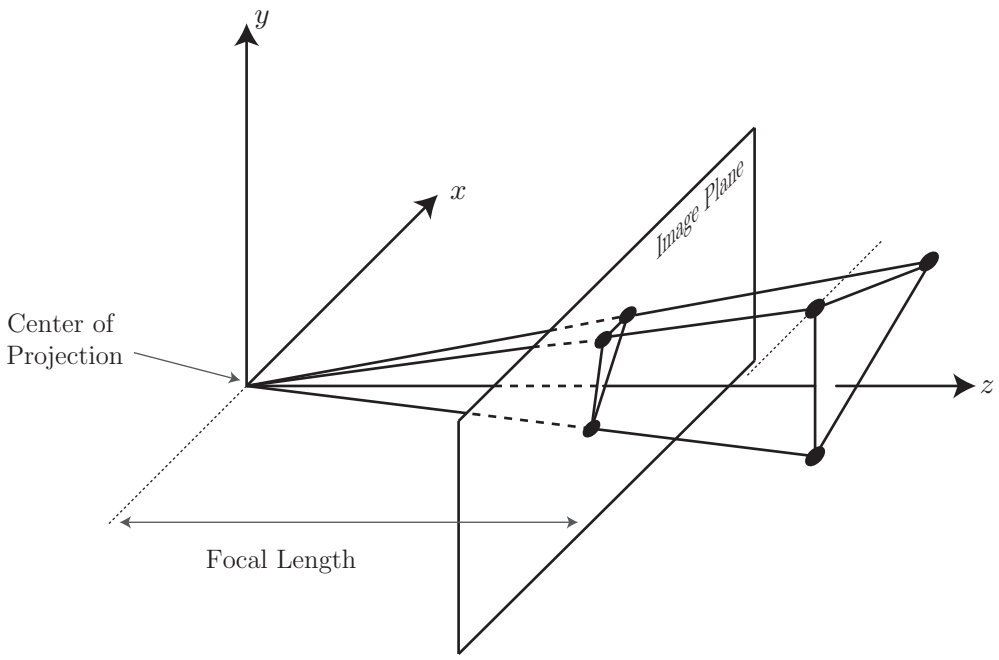


Figure 4.3 – Pinhole model.

$$x = \frac{(u - c_u)}{f_c}, \tag{4.5}$$

$$y = \frac{(v - c_v)}{f_c}, \tag{4.6}$$

$$d = \frac{1}{Z}, \tag{4.7}$$

where x and y are the coordinates of 2D points on image plane, in meters (instead of u and v which were in pixels) and d is the inverse depth. For reasons which will become clear shorty, it's preferred to work with the inverse of depths instead of depths themselves. Finally, the coordinates of several image points can be stacked together to give the feature vector, $\mathbf{s} = [x_1, y_1, x_2, y_2, \cdots]^T$, and several inverse depths are stacked to make up the vector of (inverse) depths, $\mathbf{d}$.

4.1.3 Dynamics of Features

Since it's assumed here that the target object is rigid, $\mathbf{s}$ is a function of ${}^b\mathcal{F}_c$ and ${}^b\mathcal{F}_o$, which are the camera and object coordinate frames, respectively. Mathematically,

$$\mathbf{s} = \mathbf{s}(\mathcal{F}_c, \mathcal{F}_o). \tag{4.8}$$

In an eye-in-hand configuration, for instance, $\mathcal{F}_c$ is a function of the robot joints and $\mathcal{F}_o$ is a function of the movements of the object in the base frame. Also, the image features are

not directly a function of the base frame, as they only depend on the pose of the camera *relative* to the object; if the camera and the object were rotated and translated identically, w.r.t. the base frame, the image and therefore the image features wouldn't change. Therefore, Eq. 4.8 can be written more accurately as,

$$\mathbf{s} = \mathbf{s}\left({}^{c}\mathcal{F}_{o}\right). \tag{4.9}$$

Furthermore, it is possible to establish a linear relationship between the time variation of $\mathbf{s}$ and the time variation of ${}^{c}\mathcal{F}_{o}$, represented in velocity screw form, as explained in [118, 119].

$$\dot{\mathbf{s}} = \mathbf{L}_{\mathbf{s}}\left[{}^{c}\boldsymbol{\nu}_{c}\left(\dot{\mathbf{q}}\right) - {}^{c}\boldsymbol{\nu}_{o}\left(t\right)\right]. \tag{4.10}$$

The first term in the parenthesis is the velocity screw of the camera represented in the camera coordinate frame and the second term is the contribution of the motion of the object. In this chapter, it is assumed that the target object is static (or quasi-static). This means that the there is no (or little) contribution from the motion of the object. Therefore,

$$\dot{\mathbf{s}} = \mathbf{L}_{\mathbf{s}}{}^{c}\boldsymbol{\nu}_{c}. \tag{4.11}$$

$\mathbf{L}_{\mathbf{s}}$ is referred to as the *interaction matrix* or the image Jacobian. The interaction matrix can be derived by differentiating the projection Eqs. 4.2–4.4. Therefore, for a single 2D point feature, the interaction matrix is given by,

$$\mathbf{L}_{i} = \begin{bmatrix} -d_{i} & 0 & x_{i}\cdot d & x_{i}\cdot y_{i} & -\left(1+x_{i}^{2}\right) & y_{i} \\ 0 & -d_{i} & y_{i}\cdot d_{i} & 1+y_{i}^{2} & -x_{i}\cdot y_{i} & -x_{i} \end{bmatrix}. \tag{4.12}$$

For more features, the interaction matrix is given by stacking the above matrix for each 2D feature. That is,

$$\mathbf{L}_{s}\left(\mathbf{s},\mathbf{d}\right) = \underbrace{\begin{bmatrix} \mathbf{L}_{1} \\ \mathbf{L}_{2} \\ \vdots \\ \mathbf{L}_{n_{f}} \end{bmatrix}}_{\left(2n_{f}\times 6\right)} \tag{4.13}$$

where n_f is the number of features. The interaction matrix is a function of the feature vector, $\mathbf{s}$, and the vector of feature depths, $\boldsymbol{d}$. Given the above, the rate of change of the features can be related to the velocity of the camera by,

$$\underbrace{\dot{\mathbf{s}}}_{\left(2n_{f}\times 1\right)} = \underbrace{\mathbf{L}_{s}}_{\left(2n_{f}\times 6\right)}\underbrace{\begin{bmatrix} {}^{c}\mathbf{V}_{c} \\ {}^{c}\Omega_{c} \end{bmatrix}}_{\left(6\times 1\right)}. \tag{4.14}$$

Since the camera is rigidly attached to the end-effector, the velocity of the camera can be related to the time variation of robot joints, $\dot{\mathbf{q}}$. Indeed, one can relate ${}^{c}\boldsymbol{\nu}_{c}$ and $\dot{\mathbf{q}}$ through the robot Jacobian as,

$$ {}^{c}\boldsymbol{\nu}_{c} \;=\; \begin{bmatrix} {}^{c}\mathbf{V}_{c} \\ {}^{c}\Omega_{c} \end{bmatrix} = {}^{c}\mathbf{J}_{r}\left(\mathbf{q}\right)\dot{\mathbf{q}}. \tag{4.15} $$

It's of practical importance to pay attention to the coordinate frames in which these quantities are defined. In Eq. 4.15, ${}^{c}\mathbf{J}_{r}\left(\mathbf{q}\right)$ is the robot Jacobian referenced in the camera frame. In practice, either the Jacobian of the robot in the base frame has to be transformed to the camera frame or the velocity screw of the camera has to be transformed into the base frame using Eq. 4.1. Therefore, represented in base frame, one has,

$$ \boldsymbol{\nu}_{c} = \mathbf{J}_{r}\left(\mathbf{q}\right)\dot{\mathbf{q}}. \tag{4.16} $$

where $\mathbf{J}_{r}\left(\mathbf{q}\right)$ and $\boldsymbol{\nu}_{c}$ are represented w.r.t the base frame[1]. Therefore, the relation between time variation of the feature vector and the robot joint velocities is given by,

$$ \dot{\mathbf{s}} = \mathbf{L}_{\mathbf{s}}\left(\mathbf{s},\mathbf{d}\right)\mathbf{J}_{r}\left(\mathbf{q}\right)\dot{\mathbf{q}}. \tag{4.17} $$

Indeed, Eq. 4.17 is the starting point in many works, such as the tutorial in [107].

4.1.4 Classical vs. Advanced Control Methods and Problem Definition

An important classical method in visual servoing is known as *image-based visual servoing* (IBVS), in which, in order to drive the robot into the desired pose, some visual features are chosen in an initial image and a target image, and the goal of the control is set to align these two sets of features. It is possible to show that for sets of four or more features, if the error between all 2D features in the initial image and the target image are reduced to zero, the resulting pose is the exact pose from which the target image was observed [118]. Therefore, IBVS tries to reach the desired pose, indirectly, through minimizing the error between a set of observed 2D features and a set of target 2D features. All methods that try to reduce feature errors in *image space,* in order to reach a target pose, are referred to as image-based methods. Fig. 4.4 illustrates the idea behind the IBVS.

Another important classical method is known as *position-based visual servoing* (PBVS), in which, the pose of the robot is *directly* regulated . Naturally, position-based approaches require an estimation method for determining the current pose of the robot with respect to the target. For performing *pose-estimation*, position-based methods often need more information about the environment and the object, compared to the image-based methods [109]. For instance, a lot of PBVS approaches require a 3D model of the target object for

[1]The superscript, b, has been dropped by convention, as previously stated.

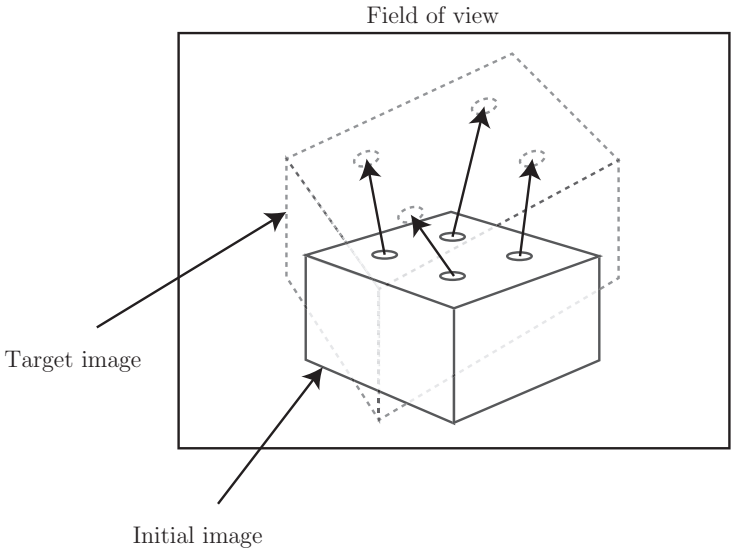


Figure 4.4 – Image-based visual servoing. The 2D features in the initial image are matched and aligned with the 2D features in the target image, and as a result, the robot is moved to the desired pose.

accurate pose estimation. Even so, accurate pose estimation is often quite demanding and modeling errors and measurement noise can result in pose estimates which are far from accurate. As a result, IBVS methods are commonly believed to be more robust to modeling errors and other forms of uncertainty [105, 109]. However, while PBVS can be a straightforward problem in control level, IBVS is more challenging, because of the highly coupled and nonlinear dynamics of the problem, in addition to uncertainties in the depth information.

In this chapter, an image-based methodology is to be perused, because of the smaller amount of information needed for implementation of IBVS, compared to PBVS. Specifically, it is not desirable to assume that a 3D model of the target object is always available. Therefore, in this chapter, minimal assumptions are made about the requirements of the problem.

The standard IBVS method tries to regulate the error between the desired values of the features and their current values, through a simple proportional type controller of the form,

$$\dot{\mathbf{q}} = -\lambda \left(\mathbf{L}_{\mathbf{s}}\mathbf{J}_r\right)^{\dagger} \left(\mathbf{s} - \mathbf{s}^{\star}\right), \tag{4.18}$$

where, $\mathbf{s}^{\star}$, is the desired values of the features, $'\dagger'$ is the pseudo-inverse operator, and λ is the gain for the proportional controller. In fact, when λ is equal to 1, the above control law is equivalent to a one-step optimal controller, which tries to minimize the following quadratic cost[2],

[2]When λ is not equal to one, the cost function has a slightly different form:$c\left(\mathbf{s}\right) = \left(\mathbf{s} - \left((1-\lambda)\mathbf{s} + \lambda\mathbf{s}^{\star}\right)\right)^T \left(\mathbf{s} - \left((1-\lambda)\mathbf{s} + \lambda\mathbf{s}^{\star}\right)\right)$, which simply means instead of trying to regulate *all* of the feature errors to zero in one step, a portion of it is regulated.

$$c(\mathbf{s}) = (\mathbf{s} - \mathbf{s}^\star)^T (\mathbf{s} - \mathbf{s}^\star), \tag{4.19}$$

subject to the dynamics of the feature described by Eq. 4.17. Therefore, the classical IBVS can be though of as an optimal controller with a one-step control horizon. Because of the fact that the classical IBVS only foresees a single time step into the future, it suffers from a myopic behavior, which in turn, results in several well-documented problems [113, 120]. These problems include infinite retreat of the camera in face of large rotations, getting trapped in local minima, and singularity of interaction matrix. Also, the work-space path traveled by classical methods is not *optimal,* because there is no procedure in place for a choosing a shortest path. In fact, the coupling between translation and rotation causes the work-space trajectories to curve considerably for scenarios with large rotations.

Optimal control methods, on the other hand, look further than a single step, and therefore, are able to devise a longer plan which results in a more forward-looking behavior. As a result, large rotations and local minima can be handled with ease, and the resulting work-space paths traveled by robot using optimal control methods is shortened [111]. Indeed, there already exist several works which apply a powerful optimal control-based methodology, known as *model predictive control,* to visual servoing [110, 111, 113]. These works are fairly successful in demonstrating the strengths of optimal control for visual servoing. However, in most of the existing works, the optimization problem encountered when applying optimal control to visual servoing, is solved using general purpose nonlinear programming tools. In fact, solving a high-dimensional nonlinear program in real-time, for a reasonably large control horizon, is far from practical, even with the current computational power of computers. Therefore, all existing verification of these methods is done through computer simulations.

Thus, despite the fact that optimal control is a very desirable approach for visual servoing, without the possibility of an actual real-time implementation, these methods will stay on paper. Fortunately, in Chap. 3, a computationally efficient algorithm for solving SOC was developed, which is a good candidate for realizing optimal control for visual servoing. In the following sections, Algorithm 3.1 will be tailored specifically for IBVS and successfully applied in practice.

However, before ending this discussion, it should be acknowledged that some older works exist which apply LQR theory to IBVS, in real-time [121]. However it's been shown that LQR control for IBVS is very sensitive to modeling errors and, especially, various delays in the system. The work reported in [122] demonstrates how delays as low as a single camera frame can render LQR useless, or even unstable. In fact, in preliminary studies for this thesis, LQR was implemented in both simulations and real-time experiments. It was observed that despite very fascinating performance of LQR in simulations, in practice the system only demonstrated unstable behavior. Therefore, LQR, while interesting and potentially useful for very low frame rates, proves too sensitive for an actual use. This might be one of the reasons why LQR, despite its simplicity and theoretically good performance, hasn't become

a standard method in visual servoing.

In order to be able to apply the proposed SOC solution to IBVS, the first step is to derive a discrete-time state-space model of the system.

4.2 State Space Model for Image Features

In order to derive a state space model, the first step is to define the control signal and the state vector. For the control signal, choosing robot joint velocities is a good option, provided that a low-level joint velocity controller is readily available. In our case, a PI controller for controlling the joint velocities is already implemented and tested. Therefore, the joint velocities are taken as the control signal,

$$\mathbf{u} \triangleq \dot{\mathbf{q}}. \tag{4.20}$$

The state vector should contain the feature vector, the vector of feature (inverse) depths, and the robot joint vector. Therefore,

$$\mathbf{x} \triangleq \begin{bmatrix} \mathbf{s} \\ \mathbf{d} \\ \mathbf{q} \end{bmatrix}. \tag{4.21}$$

Using Eq. 4.17, the time variation of the state of the system is given by,

$$\underbrace{\begin{bmatrix} \dot{\mathbf{s}} \\ \dot{\mathbf{d}} \\ \dot{\mathbf{q}} \end{bmatrix}}_{=\dot{\mathbf{x}}} = \underbrace{\begin{bmatrix} \mathbf{L_s}\left(\mathbf{s},\mathbf{Z}\right)\mathbf{J}_r\left(\mathbf{q}\right) \\ \mathbf{L_d}\left(\mathbf{s},\mathbf{d}\right)\mathbf{J}_r\left(\mathbf{q}\right) \\ \mathbf{I} \end{bmatrix}}_{=\mathbf{J}(\mathbf{x})} \underbrace{\dot{\mathbf{q}}}_{=\mathbf{u}}, \tag{4.22}$$

where the derivations of $\mathbf{L_s}$ and $\mathbf{J}_r$ were already discussed, and $\mathbf{I}$ is the identity matrix. The remaining $\mathbf{L_d}\left(\mathbf{s},\mathbf{d}\right)$ term defines the time evolution of the inverse depths w.r.t. camera velocity screw ${}^c\boldsymbol{\nu}_c$, given by,

$$\mathbf{L_d} = \underbrace{\begin{bmatrix} 0 & 0 & d_1^2 & \gamma_1 \cdot d_1 & -x_1 \cdot d_1 & 0 \\ 0 & 0 & d_2^2 & \gamma_2 \cdot d_2 & -x_2 \cdot d_2 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix}}_{n_f \text{ by } 6}. \tag{4.23}$$

Therefore,

$$\dot{\mathbf{x}} = \mathbf{J}\left(\mathbf{x}\right)\mathbf{u}. \tag{4.24}$$

A Euler discretization of the above, with time step $\triangle t$ results in,

$$\mathbf{x}_{k+1} = \mathbf{A}_k \mathbf{x}_k + \mathbf{B}_k \mathbf{u}_k, \tag{4.25}$$

where,

$$\mathbf{A}_k = \mathbf{I} + \frac{\mathbf{J}(\mathbf{x})\,\mathbf{u}}{\partial \mathbf{x}} \triangle t, \tag{4.26}$$

$$\mathbf{B}_k = \mathbf{J}(\mathbf{x}) \triangle t. \tag{4.27}$$

In the above, all required terms except $\mathbf{J(x)u}/\partial \mathbf{x}$ can be readily calculated from analytical formulas. To further improve the computational efficiency of the proposed solution for visual servoing, the analytical description for $\frac{\mathbf{J(x)u}}{\partial \mathbf{x}}$ was derived by symbolically differentiating $\mathbf{J}(\mathbf{x})\,\mathbf{u}$ w.r.t. $\mathbf{x}$ using the Matlab Symbolic Toolbox. For an example case of 4 point features, and a 6 DOF robot, this matrix is an 18 by 18 matrix (the dimension of $\mathbf{x}$ is $4 \times 2 = 8$ for each 2D point, plus 4 depth values and 6 joint values). Thus, the derivation of the state space model is complete.

4.3 Partial Observability

In Chap. 2, it was explicitly assumed that the full information of the state of the system is observed in each time-step. However, for the case of IBVS, the state of the system defined by, $\mathbf{x} \triangleq \begin{bmatrix} \mathbf{s}^T & \mathbf{d}^T & \mathbf{q}^T \end{bmatrix}^T$, is not fully observed, because the depth information is not available at all. Also, the readings of the coordinates of point features in the images are subject to image noise, as well as other forms of uncertainty caused by illumination changes and temporary occlusion of the features, among other things. The joint values of the robot are available through motor encoder readings which are, at least, subject to quantization error. Therefore, the problem of IBVS studied here is *not* fully observable.

The full solution to SOC for partial observable problems involves the concept of *dual control* [123–125], which is out of the scope of this thesis. It can be pointed out, briefly, that dual control states that the problem of estimating the state of the system through noisy observations, and the problem of optimal control of the plant are interrelated. This means that the control policy should try to improve the state estimation, at the same time as it's trying to minimize the control cost (hence the name, dual).

Fortunately, for the important LQ case, there exists an influential result known as the *separation principle* [123, 126]. The separation principle states that for LQ problems, the problem of state estimation and optimal control are not interrelated, and hence, can be performed in separate phases. In fact, dual control is rarely implemented in practice, due to the fact that the exact solution suffers from curse of dimensionality, in infinite dimensions [127–129]. However, in practice, even when the separation does not hold exactly, it is often assumed as a useful heuristic.

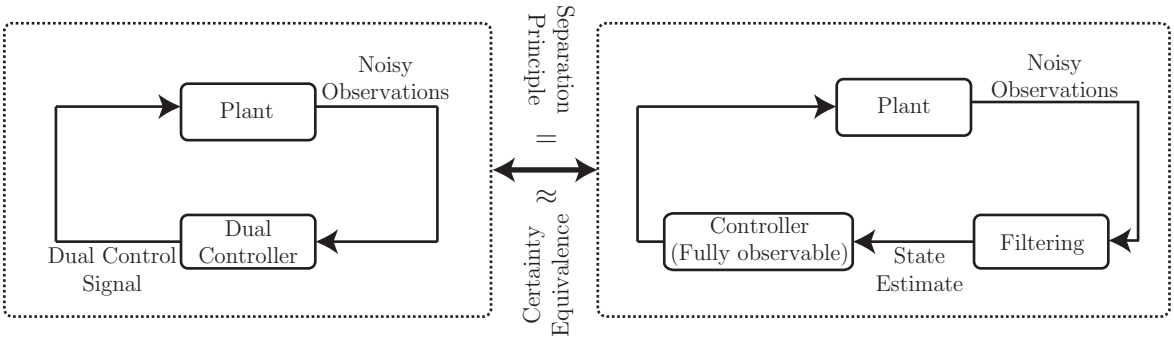


Figure 4.5 – The separation principle and certainty equivalence heuristic.

Therefore, for partially observable problems, a common approach is to break down the problem into an estimation (or filtering phase) and a control phase. The estimator outputs the most-likely state estimation, given all previous measurements. The controller, in turn, simply treats this estimation as the true value for control, and solves the control by implicitly making a full observability assumption. This assumption is sometimes known as the certainty equivalence heuristic [123, 128]. Fig. 4.5 summarizes the topology of the controller and filter when certainty equivalence is assumed.

The certainty equivalence heuristic described above is also adopted in this work. Specifically, an EKF is implemented to observe the noisy image features and motor encoder readings, and to estimate the unknown (inverse) depths of the features. The observation equations can be described by,

$$\mathbf{o}_k = \begin{bmatrix} \mathbf{s}_k \\ \mathbf{q}_k \end{bmatrix} + \mathbf{w}_k = \mathbf{H}\mathbf{x} + \mathbf{w}_k, \tag{4.28}$$

where,

$$\mathbf{H} = \begin{bmatrix} \underbrace{\mathbf{I}}_{(2n_f \times 2n_f)} & \underbrace{\mathbf{0}}_{(n_f \times n_f)} & \underbrace{\mathbf{I}}_{(n_q \times n_q)} \end{bmatrix}. \tag{4.29}$$

Given the linear approximation of the dynamics of the system described by Eq. 3.80, and using the standard EKF equations derived in Appx. D , the state, including (inverse) depth estimates can be derived.

$$\hat{\mathbf{x}}_k = \begin{bmatrix} \hat{\mathbf{s}}_k \\ \hat{\mathbf{d}}_k \\ \hat{\mathbf{q}}_k \end{bmatrix}. \tag{4.30}$$

This state estimation is updated iteratively, and fed to the controller. The reason that inverse depths were used instead of depths themselves is precisely because inverse-depth estimation with EKF is known to be more accurate than depth estimation [130–132]. The

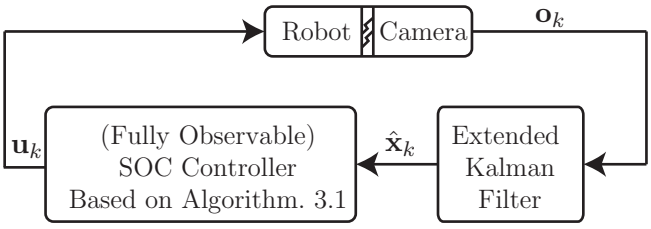


Figure 4.6 – The control topology, as a result of the assumed certainty equivalence heuristic. An EKF is used as the state estimator, and the state estimation is fed to a fully observable SOC controller.

reason can be simply summarized by the fact that the inverse of depth may be better approximated by a Gaussian distribution than the depth itself. With the adoption of such estimation technique, the topology of the controller and estimator for our visual servoing application can be depicted as in Fig. 4.6.

4.4 Cost Function

The IBVS problem was brought into standard state space model as in Eq. 3.80. Therefore, given a cost function, Algorithm 3.1 can be applied to IBVS, with minimal modification. In fact, a simple quadratic cost function may be defined as,

$$c\left(\mathbf{x}_k, \mathbf{u}_k\right) = \left(\mathbf{x}_k - \mathbf{x}_k^\star\right)^T \begin{bmatrix} \underbrace{\mathbf{Q}_k}_{(2\times n_f)\text{by}(2\times n_f)} & \mathbf{0} \\ \mathbf{0} & \underbrace{\mathbf{0}}_{(n_f+n_q)\text{by}(n_f+n_q)} \end{bmatrix} \left(\mathbf{x}_k - \mathbf{x}_k^\star\right) + \mathbf{u}_k^T \mathbf{R}_k \mathbf{u}_k,$$

where, $\mathbf{x}_k^\star = \begin{bmatrix} \mathbf{s}_k^{\star T} & \mathbf{0}^T \end{bmatrix}^T$, is the desired state vector, which contains the desired feature vector, and $\mathbf{Q}_k$ and $\mathbf{R}_k$ are positive semi-definite weighting matrices that can be defined by the control designer in order to achieve the desired behavior. The first term in the cost function above penalizes the deviation of the state from the desired state, $\mathbf{x}_k^\star$, whereas the second term penalizes large control signals.

4.5 Scheduling for Real-time Implementation

Algorithm 3.1 presented in Chap. 3 for general SOC problems was essentially an *offline* algorithm. That is, in an initial phase, with hundreds of iterations, a locally optimal policy was constructed and then, in a second phase, the policy was evaluated and executed on the plant. For a partially observable case, however, an offline solution is not possible, because the estimation of the state of the system is constantly being updated, and these updates constantly change the initial point of the solution.

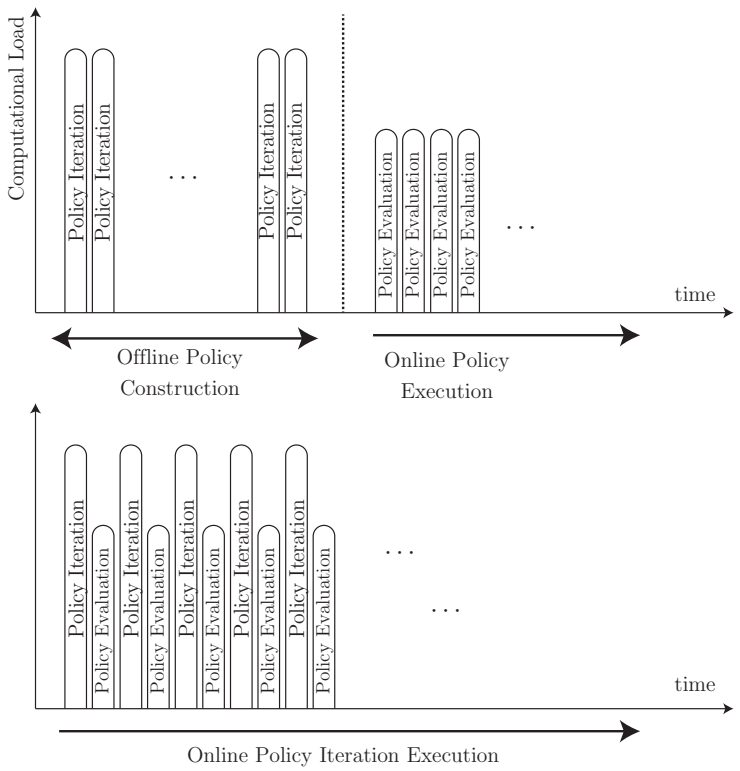


Figure 4.7 – The scheduling of computations in the original offline algorithm for SOC vs. the modified online algorithm.

One solution is to solve many problems offline, each with a possible initial condition for the problem. However, the number of such offline problems grows exponentially w.r.t. the state dimension. Therefore, for a large state space problem such as visual servoing, an offline solution is not possible. Moreover, in a single frame, even with a relatively small control horizon, the number of required iterations for complete convergence to a locally optimal policy is too large.

For online implementation of the proposed algorithm, one option is to spread out the policy updates in between, policy evaluations, in the execution phase. This idea comes from *model predictive control* methodology and is known as the *receding horizon* technique. In fact, as a future research direction, the authors of iLQG method [61] propose that a receding horizon technique together with an EKF be developed for application to partially observable problems. This is exactly what is pursued in the rest of this chapter, for application to visual servoing.

With receding horizon technique, at each execution step, an optimal control problem is solved from scratch, and the *first* control signal of the resulting control sequence is applied to the plant. In the next execution step, the optimal control is solved from the top. Fig. 4.7 depicts the modified policy update and execution schedules for realizing an online version of Algorithm 3.1.

As a comparison of the above method with MPC, it should be noted that in conventional

MPC, the optimal control problem solved in each iteration is an *open-loop*, *deterministic* problem. However, in the proposed method here, each backward policy update solves a locally *optimal policy* problem. Therefore, the proposed method here essentially constitutes a *closed-loop* MPC method, such as the one described in [133]. The close-loop MPC methods are known to be superior to the conventional open-loop MPCs, in terms of robustness [133]. Therefore, in principle, the proposed algorithm here is stronger than the existing optimal-control-based methods for visual servoing, which all based on conventional MPC [110, 111, 134]. Unfortunately this fact cannot be easily verified in practice, because existing works using conventional MPC for visual servoing are simply too computationally prohibiting to be implemented for comparison.

4.6 The Choice of α_k and β_k Parameters

Before Algorithm 3.1 can be applied to visual servoing, the choice of the parameters α_k and β_k^{-1} should be clarified. For the application to visual servoing, these parameters are all chosen to be zero. The reason is that, for the particular computational resources that is available, adding robustness to solution requires additional iterations to reach convergence. It will be explained that the maximum number that the policy is updated is equal to the horizon plus one. Therefore, In order to reach convergence for robust case, one needs to increase the control horizon. Increasing the horizon, however, also increases the computational load.

For the time being, since real-time applicability is the goal, the choices of zero for these parameters is justified. However, if more computational power was available, one could also experiment with different risk-seeking or robust variations of the proposed algorithm. With this specific choice for the set of parameters α_k, β_k^{-1}, from Chap. 3 it is known that, Algorithm 3.1 results in a local solution to standard SOC, and the policy exactly matches the one given by iLQG. However, since iLQG has not ever been used for visual servoing either, the proposed algorithm for visual servoing, described in the following section, remains novel.

4.7 The Description of the Complete Algorithm

Each iteration of the proposed algorithm for visual servoing consists of 5 steps. The first step is to receive an image from the camera and search it for visual features (distinctive blubs, in our case). The centroids of the blubs are then extracted to serve as the set of 2D point features. Also, the robot joint encoders are read to find the current values of the robot joints. The set of features and the joint values are stacked into the observation vector, $\mathbf{o}_k$.

The second step is to update the state estimation, by passing this new observation vector, $\mathbf{o}_k$, together with the previous joint velocity commands, $\mathbf{u}_{k-1}$, to the EKF. The EKF finds the most likely approximation of the state of the system, $\hat{\mathbf{x}}_k$, and passes it to the optimal

controller. The optimal controller treats $\hat{\mathbf{x}}_k$ as the true value for $\mathbf{x}_k$.

The optimal controller is based on Algorithm 3.1. Therefore, the third step is a forward simulation of the dynamics. Thus, after receiving the state, $\mathbf{x}_k$, the controller applies the most updated policy, $\left\{\mu_k^{(i)}, \mu_{k+1}^{(i)}, \cdots, \mu_{k+N}^{(i)}\right\}$, to the linearized dynamics of the system, to compute a predicted trajectory. In this step, the controller essentially computes what would happen if the current policies were to be applied to the system. The result of this step is a predicted control-state trajectory, $\{\mathbf{x}_k^{(i)}, \mathbf{u}_k^{(i)}, \mathbf{x}_{k+1}^{(i)}, \cdots \mathbf{x}_N^{(i)}\}$ and the linearized dynamics around this trajectory.

The fourth step is to update the policies. This is done in a backward sweep starting from the end of the horizon, $\mu_{k+N}^{(i)}$, and applying C.25–C.30 and C.48–C.55 , to get the updated policy $\mu_{k+N}^{(i+1)}$. This is done for all decisions, one by one, and backward in time. The result is the updated and improved policy, $\left\{\mu_k^{(i+1)}, \mu_{k+1}^{(i+1)}, \cdots, \mu_{k+N}^{(i+1)}\right\}$. It should be noted that with α_k, β_k^{-1} chosen to be zero, this backward sweep is very similar to a Riccati recursion, similar to the one in a backward pass of iLQG [61].

The fifth and final step is to pop out the first decision, $\mu_k^{(i+1)}$, and execute it by picking a sample from $\mathbf{u}_k \leftarrow \mu_k^{(i+1)}(\mathbf{u}_k \mid \hat{\mathbf{x}}_k)$, and applying it to the system. Algorithm 4.1 summarizes the above in a pseudo-code. The difference between the online solution proposed here and the offline version in Algorithm 3.1, will be clear by revisiting the last step. In the offline algorithm, all $\mu_0^{(i)}$ through $\mu_N^{(i)}$, were updated hundreds of times offline, and then this updated policies, $\left\{\mu_0^{(500)} \cdots \mu_N^{(500)}\right\}$, was applied to the system. However, in the online version, at time step zero, after only one update, the policy ,$\mu_0^{(1)}$, is applied to the system. At the next time step, the applied policies is $\mu_1^{(2)}$, and so on. Since the control horizon is N, each decision can be updated at most $(N+1)$ times, before it's time to execute it. Therefore, in the online version, the policy applied to the system is $\left\{\mu_0^{(1)}, \mu_1^{(2)}, \cdots, \mu_N^{(N+1)}, \mu_{N+1}^{(N+1)}, \mu_{N+2}^{(N+1)}, \cdots\right\}$. This means, in order for the policies to be near optimal, the control horizon has to be chosen as large as possible. In the real-time implementation discussed in the next section, a control horizon of 25 steps was achieved.

Finally, a block diagram illustrating the components of the proposed algorithm is given in Fig.4.8.

Algorithm 4.1 The pseudo-code of the proposed algorithm for visual servoing.

while !Terminated **do**
  $\mathbf{o}_k \leftarrow$Extract the feature points and read joint encoders
  $\hat{\mathbf{x}}_k \leftarrow \text{EKF}\left(\mathbf{o}_k, \hat{\mathbf{x}}_{k-1}, \mathbf{u}_{k-1}, \Sigma_{k-1}\right)$ # Update State Estimation.
  $\mathbf{x}_k \leftarrow \hat{\mathbf{x}}_k$ # Certainty Equivalence Heuristic.
  for all $k' \in \{k \cdots k+N\}$. # Forward Simulation. Compute a predicted trajectory. **do**
    $\mathbf{u}_k^{(i)} \xleftarrow{rand.} \mu_k^{(i)}\left(\mathbf{u}_k \mid \mathbf{x}_k\right) = \mathcal{N}\left(\mathbf{u}_k \mid \mathbf{K}_k^{(i)}\mathbf{x}_k + \mathbf{u}_k^{ol(i)}, \Sigma_{\mu_k^{(i)}}\right)$.
    $\mathbf{x}_{k+1}^{(i)} \leftarrow \mathbf{A}_k\left(\mathbf{x}_k\right)\mathbf{x}_k^{(i)} + \mathbf{B}_k\left(\mathbf{x}_k\right)\mathbf{u}_k^{(i)}$.
  for all $k^{'} \in \{k+N \cdots N\}$ # Backward Pass. **do**
    $\left\{\mathbf{K}_k^{(i+1)}, \mathbf{u}_k^{ol(i+1)}, \Sigma_{\mu_k^{(i+1)}} \mid k \in \{0 \cdots N\}\right\} \leftarrow$Update policies using Eqs. C.25–C.30 and C.48–C.55
    given, $\mathbf{A}_k\left(\mathbf{x}_k\right)\mathbf{x}_k, \mathbf{B}_k\left(\mathbf{x}_k\right), \mathbf{Q}_k, \mathbf{R}_k, \mathbf{x}_k^{\star}$
  $\mathbf{u}_k \leftarrow \mu_k^{(i+1)}\left(\mathbf{u}_k \mid \mathbf{x}_k\right)$
  $k++$ # Time Passes.

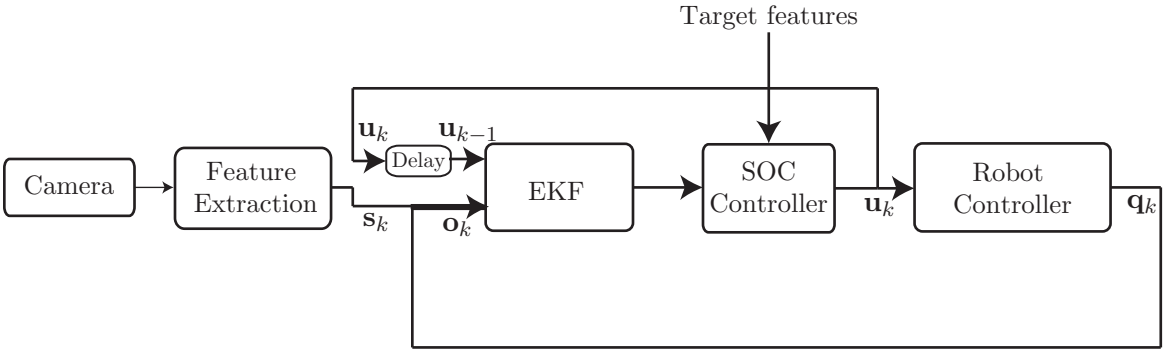


Figure 4.8 – A block diagram depicting the proposed algorithm for visual servoing.

4.8 Experimental Setup

For our experiments, a 5 DOF robotic manipulator (CRS-A255 [135]) is used. In order to verify the applicability of the control methodology in real-time, the CRS-A255 robot was controlled through Simulink, using the Matlab Real-time Workshop Toolbox and via a QuaRC Windows Target PCI board [136]. An industrial digital CMOS camera (FireFly MV [137]) with a resolution of 640×480 and frame rate of 60 Hz is attached to the end-effector of the robot.

The first phase of preparing the setup is finding the parameters of the pinhole model, through camera calibration. The is done using the Matlab Camera Calibration Toolbox, which uses the method reported in [117]. The result of the camera calibration phase is a set of values for parameters of the pinhole model, listed in Table. 4.1.

Of course, there is always some image distortion, depending on the type of the lens, which does not allow us to use a simple pinhole model, without modeling the distortion. Therefore, a distortion mapping model is also given by the calibration procedure. Using the inverse of this distortion model, it is possible to correct the effect of distortion. After distortion is removed, the pinhole model assumption can be safely assumed. Fig. 4.9 shows some images obtained from the camera, before and after applying the inverse distortion mapping.

The second phase is deriving the robot Jacobian. This step is a basic procedure in robotics, for which, one needs to form the table of Denavit–Hartenberg (DH) parameters (Table. 4.2). This table is based on the nominal values available in the manual of the robot [135], and may be slightly inaccurate. However, for our experiments, these nominal values were used. Having this information, it is possible to find the robot Jacobian using basic results on differential kinematics, which can be found in all robotics textbooks [114].

To establish ground truth for poses, a high accuracy optical tracking system (NDI Optotrack Certus[138]) is employed. The optical tracker can track the 3D position of LED markers with an accuracy of 0.1 mm. Four markers are placed on the target object such that they are visible by both camera and the optical tracking device. A coordinate frame is defined with respect to these LED markers, and the pose of target is tracked accurately using the tracking device. The same makers are also detected as blubs in images obtained from the camera, and used in our visual servoing algorithm as 2D visual features. Therefore, the accuracy of depth estimation can be verified using data from optical tracker. Also, three LED markers are placed on the end-effector and a coordinate frame is defined with respect

Parameter	Description	Value [px]
c_u	The x coordinate of the center of the image plane.	323.9
c_u	The y coordinate of the center of the image plane.	235.5
f_c	Focal length of the camera.	613.3

Table 4.1 – Values of camera parameters resulted from the calibration phase.

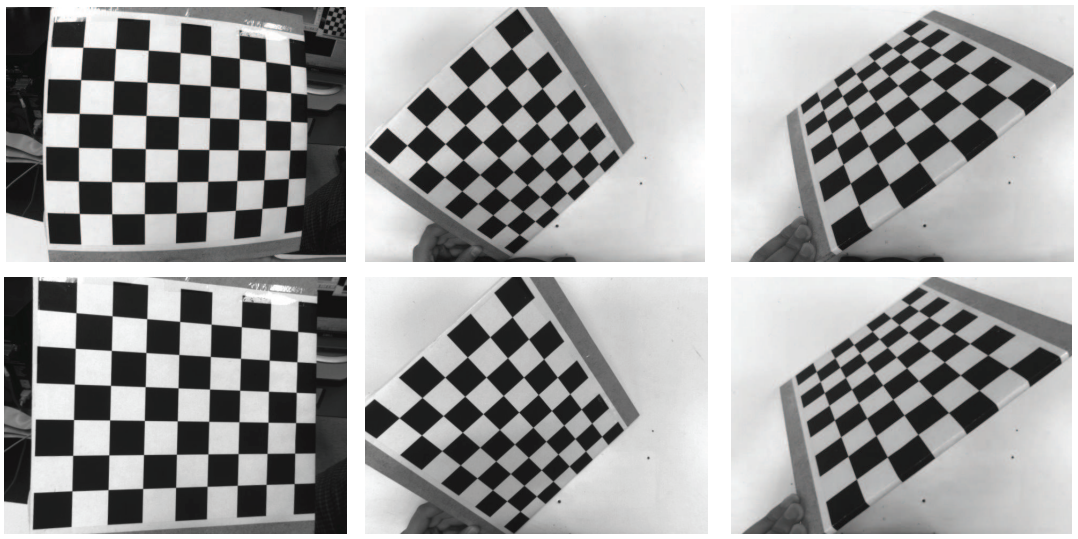


Figure 4.9 – Applying the inverse of distortion mapping. The first and second row illustrate the images obtained from the camera, before and after distortion removal, respectively.

Joint	Type	θ	d	r	α	σ
1	r	0	0.254	0	$\pi/2$	0
2	r	0	0	0.254	0	0
3	r	0	0	0.254	0	0
4	r	0	0	0	0	0
5	r	0	0.12	0	0	0

Table 4.2 – Table of DH parameters for the A255 robot.

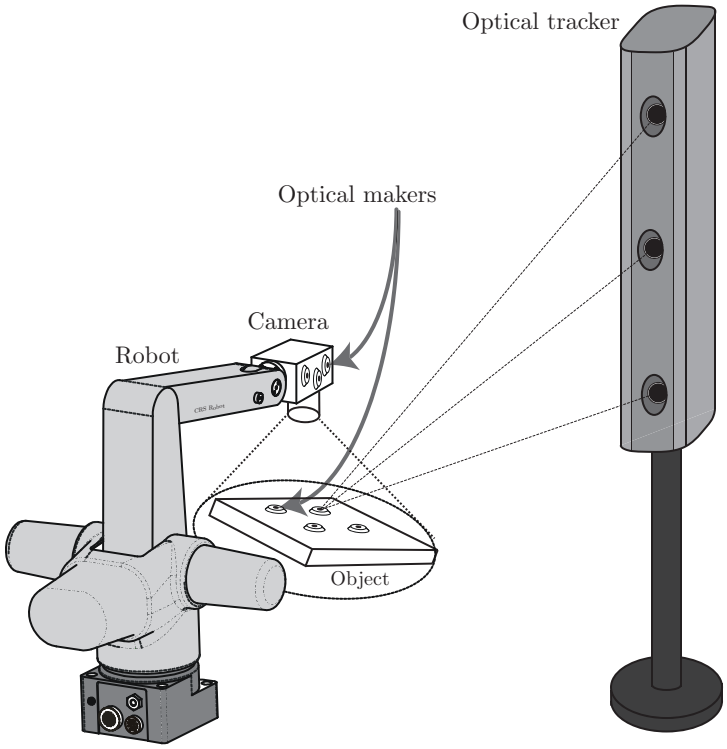


Figure 4.10 – Experimental setup. The LED markers are tracked simultaneously by the camera and the optical tracker.

to these markers. Fig. 4.10 demonstrates the experimental setup.

Each experiment follows the steps below:

1. The robot is moved to a desired pose with respect to the object.
2. The relative pose of the object with respect to the end-effector is recorded using the optical tracker.
3. A single image is recorded which constitutes our target image.
4. The LED markers are located in the image using simple blub detection, and their position in the image frame is saved as target feature vector.
5. The robot is moved to an initial pose with respect to the object.
6. The LED markers are detected in the current image and saved as initial feature vector.
7. The control sequence is started, while the optical tracker is recording the data. The controller tries to drive the robot towards the target pose.
8. After a final pose is reached, the control sequence and the optical tracker are stopped.

In existing works which apply optimal control (mostly, MPC methodology) to visual servoing, it is claimed that compared with standard IBVS, shorter workspace trajectories can be

achieved. It's also known that myopic behaviors such as the retreat of the camera in face of large rotations, and getting trapped in local minima can be avoided. For our experiments appropriate scenarios are devised to evaluate the following,

- The ability to dealing with the retreat problem.
- The ability to choosing shorter workspace paths.

The reason local minima are left out of experiments is that it turns out to be very difficult to artificially create a case of local minima with our setup, since the CRS-255 robot cannot move freely and with 6DOF. In order to evaluate the ability to deal with the retreat problem, a scenario with a large rotation needs to be considered. Specifically,

Scenario I: A pure 90°rotation around the axis normal to the image.

For this case, it is expected that standard IBVS will result in large and unnecessary backward movement of the camera, which can cause the robot to reach its joint limits. On the other hand, the proposed algorithm is expected to show minimal retreat.

To evaluate the quality of the workspace paths, two scenarios with moderate rotation and translation are considered. Specifically,

Scenario II: A combined large translation towards the object and a 30° rotation.

and

Scenario III: A combined large translation away from the object and 60° rotation.

For these two scenarios, it is expected that the proposed method will result in shorter workspace trajectories (and, consequently shorter time to complete the task). Finally, for in the accuracy of both approaches are compared. These three scenarios are illustrated in Fig. 4.11.

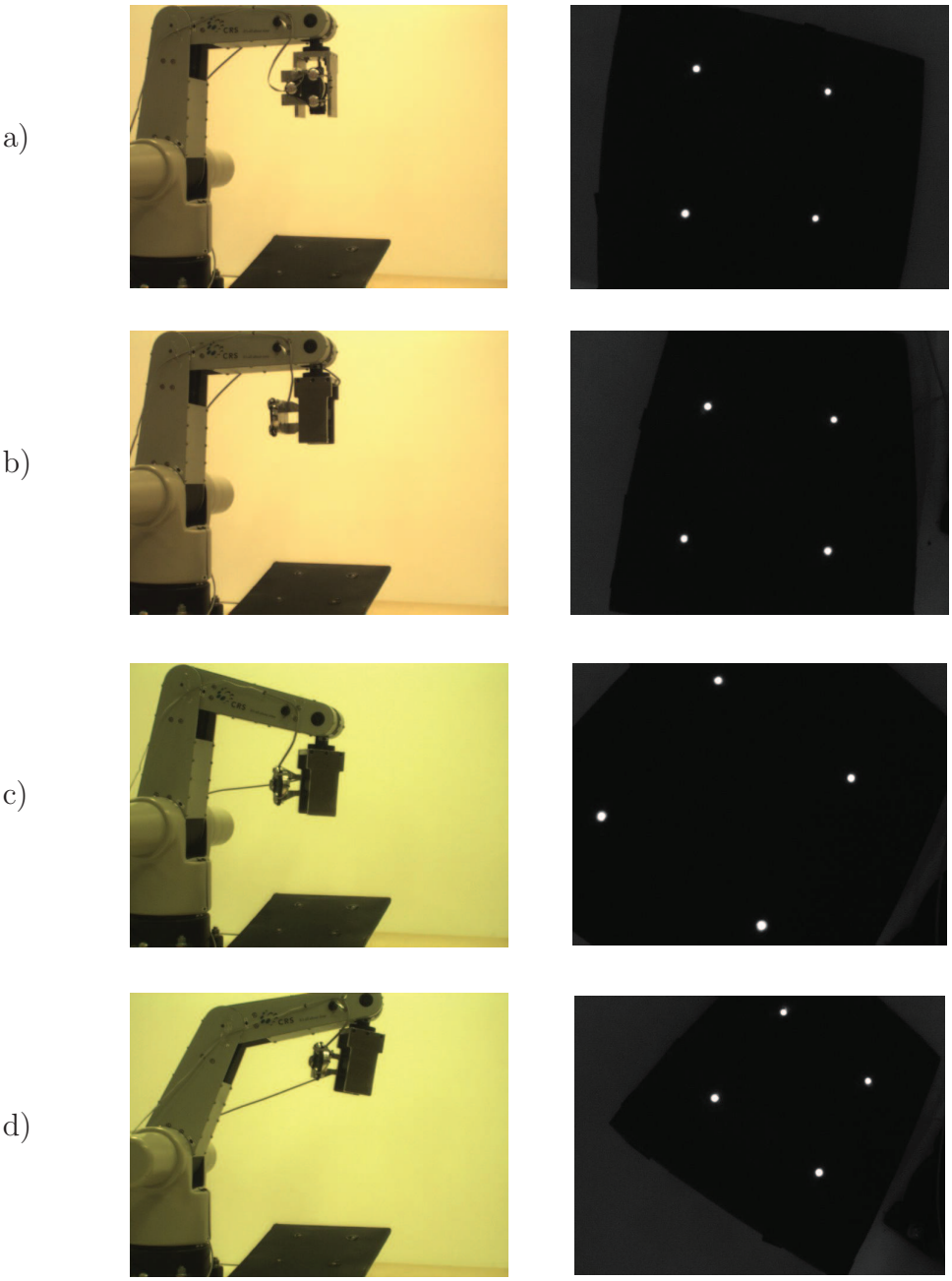


Figure 4.11 – Scenarios. Each row displays the robot pose and the corresponding image seen through the camera. a) The initial pose of the robot for all scenarios. b) The target pose for scenario I. c) The target pose for scenario II. d) The target pose for scenario III.

4.9 Experimental Results and Discussion

The performance of the proposed algorithm is compared with the standard IBVS algorithm for three scenarios. In each case, the qualitative behavior of the control as well as the orientation and translational error of the final pose and the lengths of work-space trajectories are compared.

For all of these experiments, the following parameters for the two control methodologies were used. For IBVS there is only one parameters to set,

$$\lambda = 0.1. \tag{4.31}$$

For the proposed algorithm, it was already decided to use,

$$\alpha_k = 0, \qquad \text{for all } k, \tag{4.32}$$
$$\beta_k^{-1} \rightarrow 0, \qquad \text{for all } k. \tag{4.33}$$

For the choice of weighting matrices in the cost function , the following straightforward choices were made,

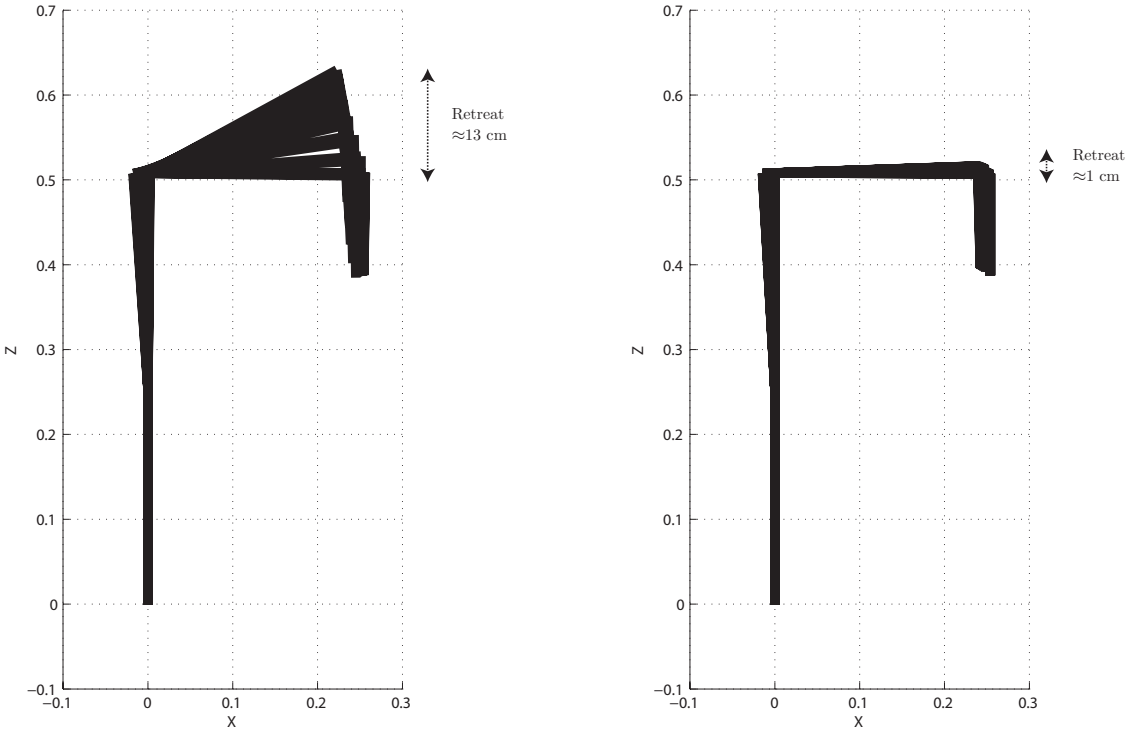
$$\mathbf{Q}_k = \mathbf{I}, \qquad \text{for all } k, \tag{4.34}$$
$$\mathbf{R}_k = 10^{-6}\mathbf{I}, \qquad \text{for all } k. \tag{4.35}$$

If a larger $\mathbf{R}_k$ was used, the input commands would be more penalized, which would result in smaller control signals. The value of $10^{-6}\mathbf{I}$ was chosen such that the control signal (which are the robot joint velocities) fall within the operation range of robot. The control horizon for the proposed method was set to 25. This is the maximum control horizon which was achievable in 60Hz.

4.9.1 Scenario I: Large Rotations and the Retreat Problem

IBVS is known to have difficulties dealing with large rotations [120]. In our experiments, it was found that even 180°rotations can be handled easily. However, since IBVS fails completely for such large rotations, for the purposes of comparison a simpler case of 90° rotation is considered. The steps described in the previous section are completed for the scenario I, for both control methodologies, and the results are collected.

Since this is a pure rotation case, there shouldn't be any translational motion. Fig. 4.12 illustrates the motion of the robot for both control methodologies. These figures are obtained by recording the joint values of the robot during the experiment, operation, and plotting the robot using the Matlab Robotics Toolbox [139]. It is observed that the standard IBVS



(a) The retreat for standard IBVS.

(b) The retreat for the proposed method.

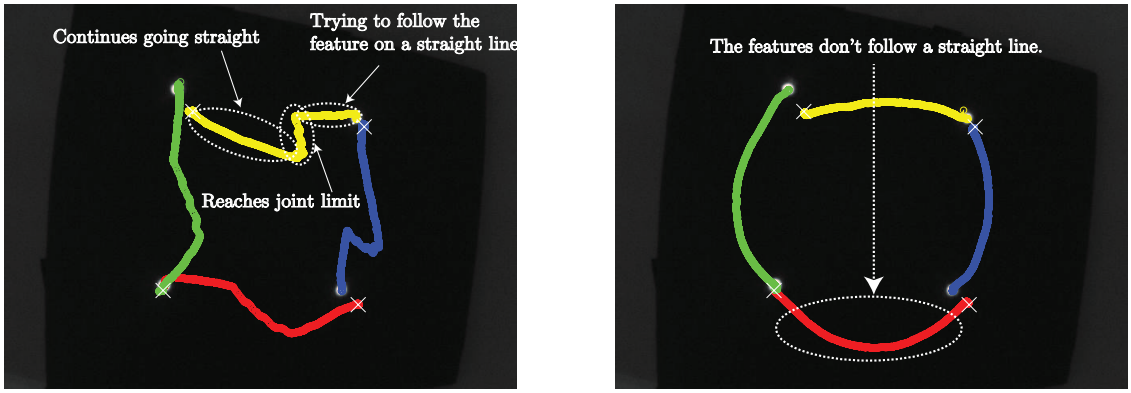
Figure 4.12 – The motion of the robot for scenario I. It is observed that the retreat is significant for IBVS.

shows a large translational motion, away from the target object.

Fig. 4.13 illustrates the trace of features in the sequence of images, during the application of both control methodologies. It can be seen that for the IBVS, the features try to move on a straight line in the image plane. However, this requires that the robot makes a large backward translation. This retreat of the camera usually results in joint limit violations, as it does in this experiment. Since one of the joints of the robots cannot move passed the joint limit, the features cannot keep following their straight path. At this point, any thing is possible. In most cases, one or many features leave the field of view, rendering the control incomplete.

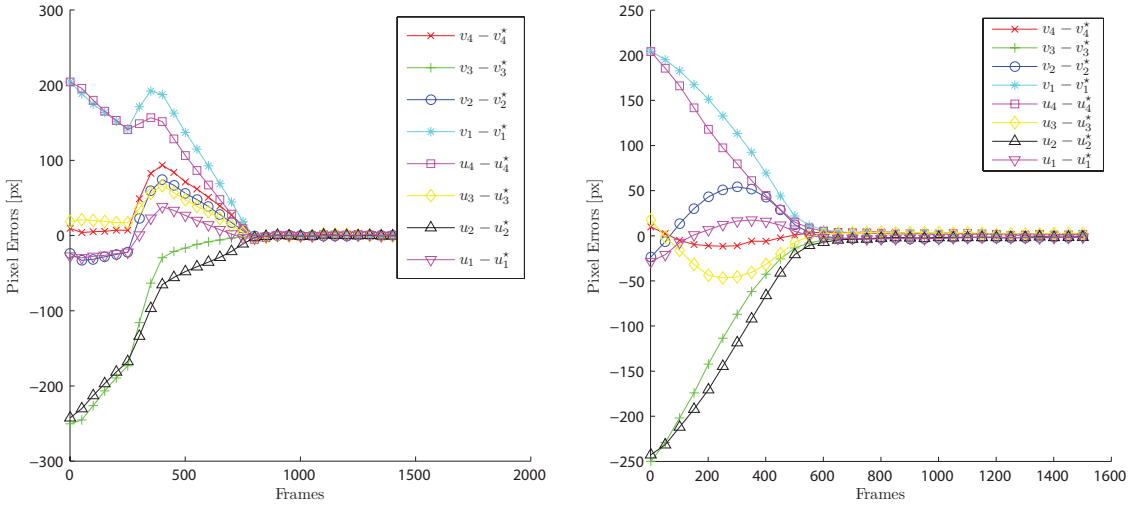
For the proposed method, on the other hand, it is seen that the image plane trajectories resemble a perfect circular motion. A comparison of the work-space trajectories presented in Table. 4.3, also shows that the trajectory length for IBVS is very large, due to very high retreat of the camera (287.0 mm compare to 37.7 mm with the proposed method). Fig. 4.14 illustrates the regulation of image feature errors to zero, by the two methodologies. Both methods have show good accuracy, regulating the features to sub-pixel levels. The orientation and translational errors of the two methods are compared in Table. 4.3, and the proposed method shows superior accuracy.

Fig. 4.15 shows the robot joint velocities for both control methodologies. For the case



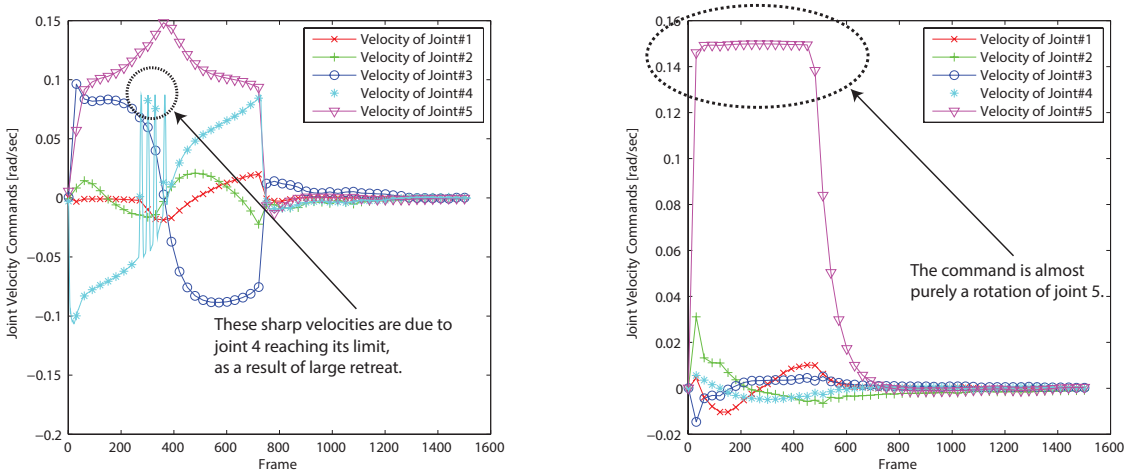
(a) The result for standard IBVS. **(b)** The result for the proposed method.

Figure 4.13 – The image space trajectories of features for scenario I.



(a) The results for IBVS. **(b)** The result for the proposed algorithm.

Figure 4.14 – The visual feature errors for scenario I.



(a) The result for IBVS.

(b) The result for the proposed method.

Figure 4.15 – The joint velocity of the robots for scenario I.

of IBVS, because of the large retreat, the robot reaches the buffer zone for joint limits. Specifically, joint 4 of the robot is pushed back into the buffer zone. In order to prevent damages to the robot, when the robot is reaches the buffer zone, a safety mechanism is activated and the violating joint is quickly pushed back into the permitted region. Therefore, the large and short-lived velocities observed in 4.15a are the result of violations of joint limits, due to large retreat. It should be noted that without this security measure, IBVS would fail as soon as the robot reached the joint limits.

Thus, the ability of the proposed method for dealing with large rotations, without the retreat problem was evaluated and verified.

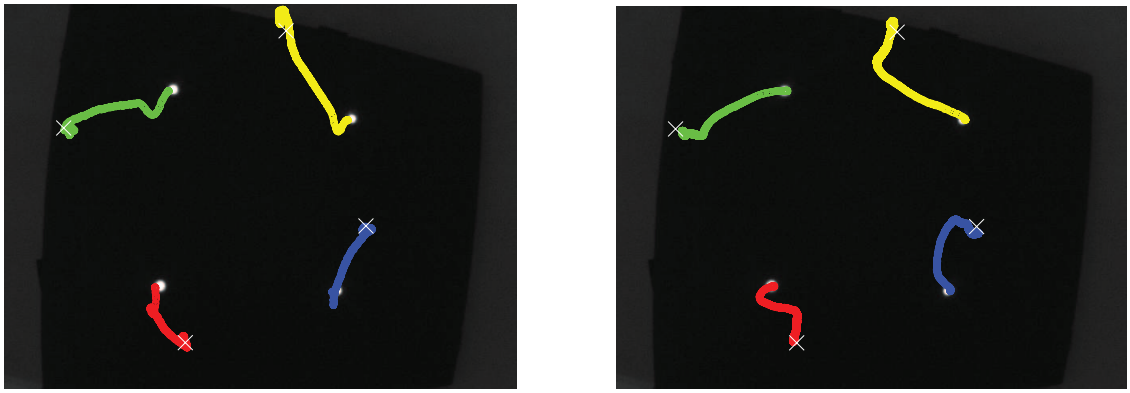
4.9.2 Scenarios II and III: Shorter Work-Space Trajectories

For scenario II and III, the goal is to show that the resulting work-space trajectories are shorter for the proposed method compare to standard IBVS. The trajectory lengths are computed by integrating the changes of the origin of the camera frame, as read from the optical tracker data. The results are summarized in Table. 4.3. According to these results, the work-space trajectory lengths for the proposed method is shorter than those of IBVS, for all scenarios. However, for cases where the rotations are larger, the trajectory length of the proposed method becomes relatively shorter. Therefore, the most improvement is seen in scenario I.

Figs. 4.16–4.21 demonstrate the image-plane trajectories, image feature errors and robot joint velocities for both control methodologies, for scenarios II and III. It is perhaps noteworthy to say point out that in Scenario III, for the proposed algorithm, one of the features leaves the field of view momentarily. This can be observed in Fig. 4.19b. Fortunately, since the EKF is implemented such that when a feature is lost, it is considered as a measurement

Scenario	Method	Orientation Error [deg]	Translational Error [mm]	Trajectory Length [mm]
I	Standard IBVS	2.05	3.4	287.0
	Proposed Algorithm	0.42	2.8	37.7
II	Standard IBVS	6.5	24.1	115.4
	Proposed Algorithm	3.5	17.8	90.2
III	Standard IBVS	4.9	24.8	122.2
	Proposed Algorithm	0.46	7.8	101.3

Table 4.3 – The comparison of translational, orientation error, and the work-space trajectory length for the three scenarios.

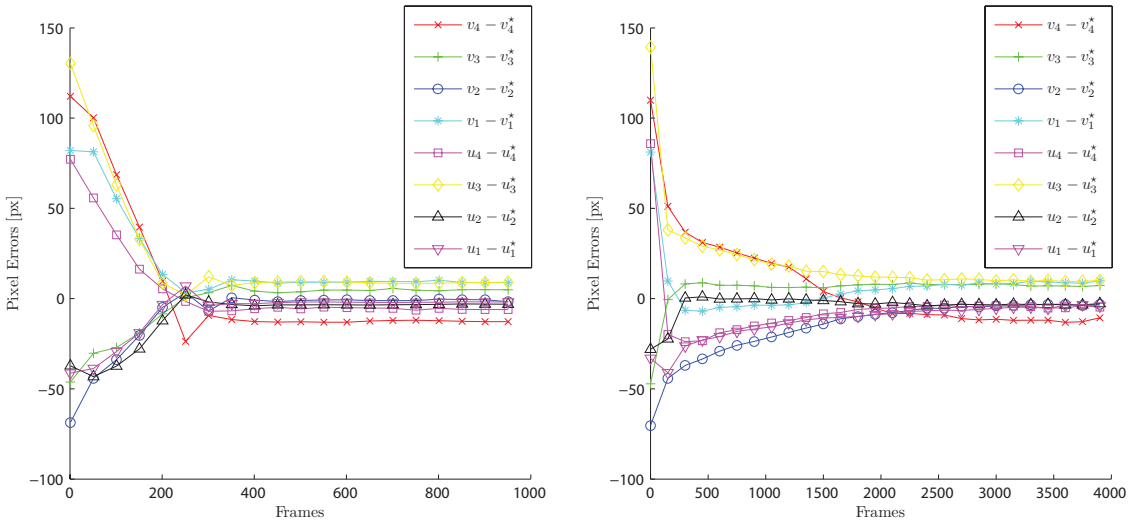


(a) The result for the IBVS.

(b) The result for the proposed method.

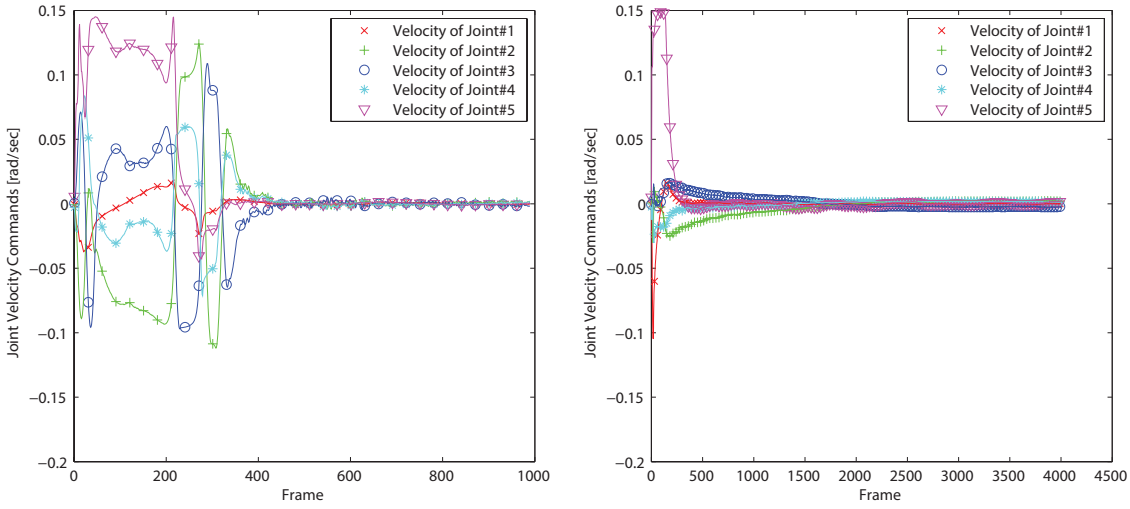
Figure 4.16 – The image space trajectories of features for scenario I.

with a variance of infinite. This technique forces EKF to output it's predicted whenever a feature is lost. Although this technique seems straightforward, and works very well in practice too, the author does not know of any works which explicitly discuss its possibility for dealing with temporary loss of features.



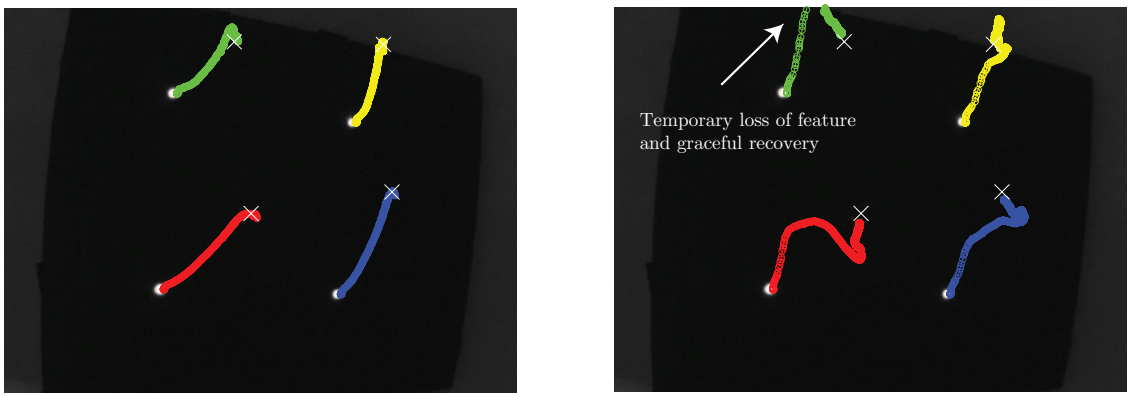
(a) The result for IBVS. **(b)** The result for the proposed algorithm.

Figure 4.17 – The visual feature errors for scenario II.



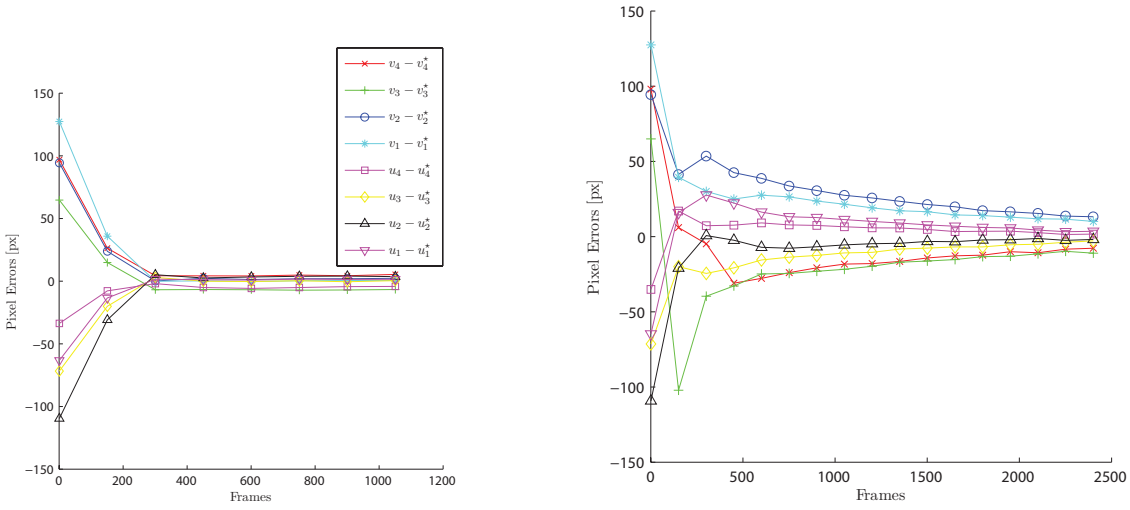
(a) The results for IBVS. **(b)** The results for the proposed algorithm.

Figure 4.18 – The joint velocity of the robots for scenario II.



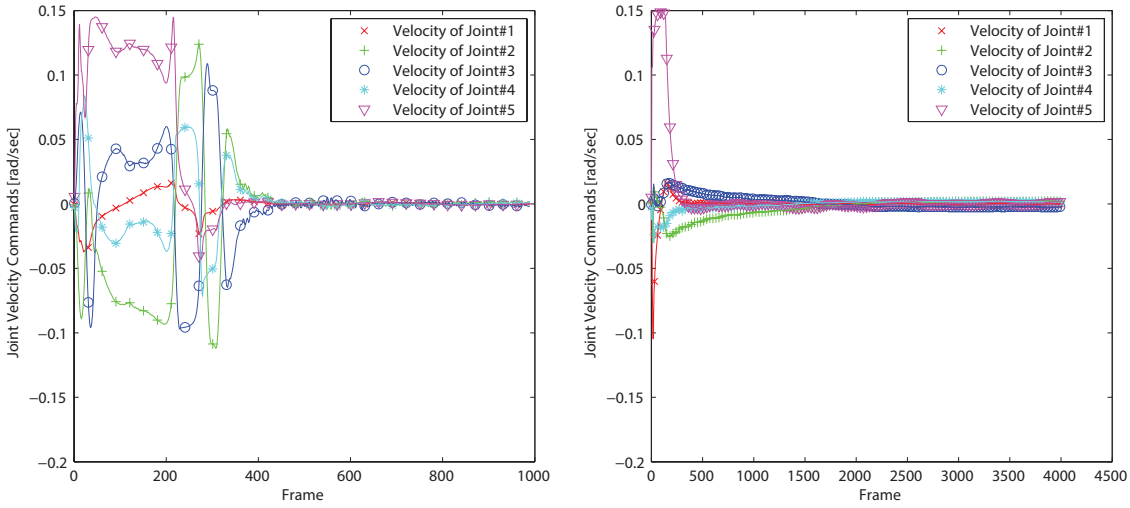
(a) The result for the IBVS. **(b)** The result for the proposed method.

Figure 4.19 – The image space trajectories of features for scenario III.



(a) The result for IBVS. **(b)** The result for the proposed algorithm.

Figure 4.20 – The visual feature errors for scenario III.



(a) The results for IBVS.

(b) The results for the proposed algorithm.

Figure 4.21 – The joint velocity of the robots for scenario III.

4.10 Summary

In this chapter, the approximate SOC algorithm developed in Chap. 3 was adapted to the problem of IBVS. The result was a powerful optimal controller for visual servoing, with modest computational load. Therefore, the algorithm was implemented in real-time, with a relatively fast frame rate of 60Hz. It was shown, through experiments and comparison to standard IBVS, that the resulting algorithm can deal with many classical problems and myopic behavior of standard IBVS, and also, result in higher control accuracy.

Chapter 5

Conclusion

5.1 Summary

The main focus of this thesis was to provide a tractable and efficient solution to SOC. The exact answer given by DP was shown to be too hard to solve for problems of interest, due to curse of dimensionality. Also, issues regarding the robustness of the optimal policy were raised. It was suggested that a promising path towards finding a tractable solution to SOC is through casting it as an inference problem, because problems in inference also suffer from curse of dimensionality, and in the context of inference, there exists a rich literature on how to deal with these computational complexities.

Therefore, a more general version of SOC was defined, which included robust, and risk-seeking SOC, and the problem was stated as an inference problem and a novel and unified solution was derived. Although this solution also suffered from curse of dimensionality, it was now possible to use approximate methods from inference. Specifically, EKF-type approximations were used and the result was an algorithm which is able to solve robust, standard, and risk-seeking SOC problems, depending on the choice of a set of parameters. This general inference based solution For a specific choice of parameters of the algorithm, the method was shown to reduce to a recent approximation to standard SOC, namely, to iLQG, and for another special case, it reduces to a recent approximate inference-based solution, with risk-seeking behavior [16]. Therefore, the proposed algorithm generalizes over these recent works, while also able to produce robust policies, therefore, completing the spectrum. Moreover, it allows for policies with time-varying behavior, which don't fit in any category of the existing methods.

For a benchmark problem, a spectrum of policies with different degrees of robustness and risk-seeking were derived. It was shown that the standard optimal policy resulted from our approach exactly match those produced by iLQG, except, in less computation time. It was also shown through simulations that compared to standard optimal policies, robust policies can reduce the cost variance significantly, while the risk-seeking policies increase it. This

means that in face of model uncertainties, the robust policies are able to accomplish the task more often than the optimal and risk-seeking policies.

For an empirical application, the developed algorithm was applied to IBVS, and the results were compared against standard IBVS methods. The computational efficiency of the proposed algorithm allows it to look ahead for a long horizon and devise an optimal plan, instead of the myopic one-step ahead methods. Also, while the existing OC-based algorithms for IBVS, cannot be used in real-time, due to their use of general nonlinear programming solvers in the loop, our efficient solution can reach long control horizons in real-time. It was shown that the proposed approach results in higher accuracy of control, shorter work-space trajectories and smaller completion time, as well as escaping local minima, and overcoming the infinite retraction problem in IBVS.

5.2 Limitations of the Proposed Methodology

The approximation based on EKF uses linearized dynamics around a predicted state-control trajectory. Therefore, it is valid in a small region around this trajectory. In fact, this is precisely why it's not subject to curse of dimensionality. However, if the uncertainties or parameter perturbations are large enough to cause the system to deviate from this trajectory, the approximate policies begin to lose their optimality. This first limitation is, indeed, inherited from EKF, and they can be lessened using more advanced types of approximations. A discussion on this will be given in the next section, under future directions.

A second limitation of the proposed approach is that the state and control constraints are not *explicitly* handled. Instead they need to be implemented as cost functions, using penalty functions, such as log-barrier penalties. However, if the feasible region of state or control space are small, using penalty functions may increase computation time. A solution for handling constraints is by using simple line-searching and backtracking techniques, also used in iLQG.

A third limitation of the propose methodology is computation time. Though much improved than previous works, it may still be too much for more complicated problems in robotics, specially those which require a long horizon to accomplish a task. In these cases, a high-level planner might be needed, or the computation time needs to be further reduced. Some ideas in this direction will be explored in the next section.

A fourth drawback of the proposed method, specifically for application in visual servoing is that, it's a lot more complicated than basic approaches in visual servoing. This is partially due to the fact that the visual servoing scenario considered here was a simple one, which could be handled with classical methods. Therefore, the proposed method could be potentially more useful when faced with more complex problem.

Finally, the proposed assumptions for achieving robust control suffer from the fact that while different degrees of robustness can be achieved, it is not exactly known what ranges

of each parameter of the model is covered. This is why considering robustness did not inflict additional computational burden. If a separate (ambiguity) range for each individual parameter was defined, the dimensions of the problem would grow by the number of uncertain parameters. Therefore, the proposed method trades computational complexity with how much control it is possible to have over what ranges of each parameter to cover.

5.3 Future Directions

There are a number of possibilities for extending the proposed solution to SOC, as well as potential applications for it. In this section, these future directions are explored.

5.3.1 Other Types of Approximations

The general proximal point solution given in Chap. 3 was approximated using EKF-type assumptions, because of the simplicity and computational efficiency of EKF. However, numerous other approximation methods are possible, given more computational resources.

For instance, approximations based on unscented Kalman filtering can be easily adapted to our work. Particle filters are also another possibility. The advantages of unscented or particle filtering over EKF also extend to the proposed method here. For instance, EKF is valid locally, in the vicinity of the trajectory around which the dynamics was linearized. Unscented, or particle filtering can result in solutions which are valid in a larger region. Also, EKF assumes Gaussian noises. Should the uncertainty be hard to approximate using Gaussian, approximations based on unscented or particle filtering can improve the accuracy of the solution. Furthermore, both unscented filtering and particle filtering are more suited for handling state or control constraints. This direction is perhaps the most direct extension of the current work.

5.3.2 Reinforcement Learning

Our solution in Chap. 3 can easily be adapted for RL. Problems discussed under RL are similar to SOC, in that, they involve a sequential decision making problem, in which, an agent needs to come up with a policy to optimize a cost or utility. The difference is that, while in SOC it was assumed that a nominal model of the plant (dynamics) and a cost function are available, in RL, it is assumed that only *samples* of the dynamics and cost function are available to the agent. The proposed algorithm can be used in the context of RL, by evaluating it at samples, and updating a functional approximation of the policy (such as a linear basis function model). This approach is similar to the one followed in [56] and [56]. For instance, a can be used to approximate the policy function, similar to [56]. The result will be a novel RL solution which has the added flexibility of being tunable for different levels of risk-seeking vs. robustness behavior.

Also, in the context of RL, risk-seeking behavior could be more useful, as it encourages learning. Therefore, a higher-level logic can be used to tune conservativeness or riskiness of the agent's behavior. Convergence speed of this proposed method is expected to be faster than existing approaches, based on the similarity of the proposed approach to other recent inference-based methods in RL, such as ψ-learning described in [16].

This direction is, perhaps, the most promising direction for future research.

5.3.3 Formal Proof of Global Convergence

Deriving a formal proof of the global convergence of our general proximal point solution was not pursued in this thesis, because ultimately, by making an EKF-type assumption, any guarantee of global convergence is lost. However, while for the purposes of this thesis developing a formal proof of convergence was not necessary or helpful, a future work in the context of RL can greatly benefit from it. In the context of RL, such a proof is necessary, because a lot of works in RL already enjoy such formal guarantees of convergence.

Fortunately, a proof can be developed by extending the results in [16]. The reason is that our solution reduces exactly to the solution in [16], for a specific choice of the parameters (when $\beta_k = \eta > 0$, for all k, which is the risk-seeking case). Therefore, for the risk-seeking case, the same proof as in [16] can be used. In the case of robust SOC, that is, (when $\beta_k < 0$), the same proof can be extended by making some assumption about the admissible range of β_k (i.e. from zero up to the breaking point).

Another possibility for deriving convergence results in by using the connections of the presented work with minimax stochastic games with relative entropy bounds and use the monotonicity properties and other useful results given in [103, 140]. This may provide a more rigorous proof than the somewhat unconventional proof given in [16]. However, using these monotonicity results may require a more rigorous reformulation of the problem within the measure theory framework, and therefore, may require some extra work. However, from a theoretical point of view, this can be a good direction with great potentials.

5.3.4 A Distributed Implementation

Similar to approximate methods in inference, a distributed implementation of our algorithm is possible. Specifically, since an asynchronous update mechanism is allowed, all messages can be computed in parallel. This can result in a much faster version of the proposed algorithm, which can be used for longer control horizons in real-time. Such an algorithm can be used for more complicated tasks which require longer plans to accomplish the task. A distributed implementation can be based on distributed and parallel implementations of message-passing algorithms for inference [141, 142].

5.3.5 Application to Advanced Robotics Problem

Since the proposed approach constitutes one of the most powerful approaches for handling nonlinear and uncertain dynamics, in large state and control spaces, it can potentially be used for hard robotics problem. For instance, motor control of humanoids may be a good area to try the propose algorithm, because of the large number of DOF of the robot, and also because of the large, continuous state space.

Appendix A

Entropy

A.1 Kullback–Leibler Divergence – or Relative Entropy

In probabiliy theory, Kullback–Leibler divergence also known as *relative entropy* is an asymmetric measure of difference between two probability distributions (or more generally, two measures on a probability space). It frequently appears in various fields such as statistical physics, information theory and variational approaches to inference [48].

Definition 1. Kullback–Leibler Divergence of probability distribution $q(x)$ *from* $p(x)$ is defined as:

$$\mathrm{KL}\left(p\left(x\right) \parallel q\left(x\right)\right) = \int p\left(x\right) \log \frac{p\left(x\right)}{q\left(x\right)} dx. \tag{A.1}$$

In general, $\mathrm{KL}\left(p \parallel q\right)$ is not equal to $\mathrm{KL}\left(q \parallel p\right)$. Also, KL-Divergence is always non-negative, with zero occuring *iff* $q\left(x\right) = p\left(x\right)$.

If $p\left(x\right)$ and $q\left(x\right)$ are unnormalized distributions, we have the more general definitions:

$$\mathrm{KL}\left(p\left(x\right) \parallel q\left(x\right)\right) = \int p\left(x\right) \log \frac{p\left(x\right)}{q\left(x\right)} dx - \int \left[q\left(x\right) - p\left(x\right)\right] dx. \tag{A.2}$$

Definition 2. The Conditional Kullback–Leibler Divergence (or conditional relative entropy) of $q\left(x \mid y\right)$ from $p\left(x \mid y\right)$ is defined as:

$$\mathrm{KL}\left(p\left(x \mid y\right) \parallel q\left(x \mid y\right)\right) = \int \int p\left(x, y\right) \log \frac{p\left(x \mid y\right)}{q\left(x \mid y\right)} dxdy = \int p\left(y\right) \int p\left(x \mid y\right) \log \frac{p\left(x \mid y\right)}{q\left(x \mid y\right)} dxdy. \tag{A.3}$$

While Definition 2 is for a bivariate case, generalizations to multivariate distributions are simple to imagine.

Appendix B

Multivariate Gaussian Functions and Distributions

We use two notations for describing normal distributions. We use two formats for representing Gaussians: A normal form, represented by parantheses and a canonical form, represented by square brackets, defined as follows,

$$\mathcal{N}(\mathbf{x} \mid \hat{\mathbf{x}}, \Sigma) = \left((2\pi)^d |\Sigma|\right)^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\hat{\mathbf{x}})^T \Sigma^{-1}(\mathbf{x}-\hat{\mathbf{x}})\right), \tag{B.1}$$

$$\mathcal{N}[\mathbf{x} \mid \check{\mathbf{x}}, \mathbf{P}] = \frac{\exp\left(-\frac{1}{2}\check{\mathbf{x}}^T\mathbf{P}^{-1}\check{\mathbf{x}}\right)}{\left((2\pi)^d |\mathbf{P}^{-1}|^{\frac{1}{2}}\right)} \exp\left(-\frac{1}{2}\mathbf{x}^T\mathbf{P}\mathbf{x} + \mathbf{x}^T\check{\mathbf{x}}\right), \tag{B.2}$$

where d is the dimension of $\mathbf{x}$, Σ is the covariance matrix and $\mathbf{P}$ is called the precision matrix. We can convert one form to the other using,

$$\mathcal{N}(\mathbf{x} \mid \hat{\mathbf{x}}, \Sigma) = \mathcal{N}\left[\mathbf{x} \mid \Sigma^{-1}\hat{\mathbf{x}}, \Sigma^{-1}\right], \tag{B.3}$$

$$\mathcal{N}[\mathbf{x} \mid \check{\mathbf{x}}, \mathbf{P}] = \mathcal{N}\left(\mathbf{x} \mid \mathbf{P}^{-1}\check{\mathbf{x}}, \mathbf{P}^{-1}\right), \tag{B.4}$$

The above definitions are well-defined if the covariance (or precision) matrix has full rank. Degenerate Gaussians are Gaussians with a singular covariance (or precision) matrix We can extend the above definitions for degenerate Gaussians as follows:

$$\mathcal{N}\left(\mathbf{x} \mid \hat{\mathbf{x}}, \Sigma_{\mathbf{x}}^{-}\right) = \left((2\pi)^d \lambda_1\lambda_2\cdots\lambda_r\right)^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\hat{\mathbf{x}})\Sigma_{\mathbf{x}}^{\dagger}(\mathbf{x}-\hat{\mathbf{x}})\right), \tag{B.5}$$

$$\mathcal{N}\left[\mathbf{x} \mid \check{\mathbf{x}}, \mathbf{P}^{-}\right] = \left((2\pi)^r \lambda_1^{-1}\lambda_2^{-1}\cdots\lambda_r^{-1}\right)^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\check{\mathbf{x}})\mathbf{P}(\mathbf{x}-\check{\mathbf{x}})\right), \tag{B.6}$$

where $\lambda_1\lambda_2\cdots\lambda_r$ are eigenvalues of $\Sigma_{\mathbf{x}}^{-}$, and $\lambda_1^{-1}\lambda_2^{-1}\cdots\lambda_r^{-1}$ are eigenvalues of $\mathbf{P}^{-}$. The superscript '†' denotes the Moore–Penrose pseudoinverse. In the following, whenever we have singular covariance (or precision) matrices we use generalized pseudoinverse and the product of eigenvalues instead of

inverse and determinant, respectively.

For the product of Gaussians we have,

$$\mathcal{N}\left[\mathbf{x} \mid \check{\mathbf{x}}_1, \mathbf{P}_1\right] \mathcal{N}\left[\mathbf{x} \mid \check{\mathbf{x}}_2, \mathbf{P}_2\right] = \tag{B.7}$$

$$\mathcal{N}\left[\mathbf{x} \mid \check{\mathbf{x}}_1 + \check{\mathbf{x}}_2, \mathbf{P}_1 + \mathbf{P}_2\right] \mathcal{N}\left(\mathbf{P}_1^{-1}\check{\mathbf{x}}_1 \mid \mathbf{P}_2^{-1}\check{\mathbf{x}}_2, \mathbf{P}_1^{-1} + \mathbf{P}_2^{-1}\right), \tag{B.8}$$

$$\mathcal{N}\left(\mathbf{x} \mid \hat{\mathbf{x}}_1, \Sigma_1\right) \mathcal{N}\left(\mathbf{x} \mid \hat{\mathbf{x}}_2, \Sigma_2\right) = \tag{B.9}$$

$$\mathcal{N}\left[\mathbf{x} \mid \Sigma_1^{-1}\hat{\mathbf{x}}_1 + \Sigma_2^{-1}\hat{\mathbf{x}}_2, \Sigma_1^{-1} + \Sigma_2^{-1}\right] \mathcal{N}\left(\hat{\mathbf{x}}_1 \mid \hat{\mathbf{x}}_2, \Sigma_1 + \Sigma_2\right), \tag{B.10}$$

$$\mathcal{N}\left(\mathbf{x} \mid \hat{\mathbf{x}}_1, \Sigma_1\right) \mathcal{N}\left[\mathbf{x} \mid \check{\mathbf{x}}_2, \mathbf{P}_2\right] = \tag{B.11}$$

$$\mathcal{N}\left[\mathbf{x} \mid \Sigma_1^{-1}\hat{\mathbf{x}}_1 + \check{\mathbf{x}}_2, \Sigma_1^{-1} + \mathbf{P}_2\right] \mathcal{N}\left(\hat{\mathbf{x}}_1 \mid \mathbf{P}_2^{-1}\check{\mathbf{x}}_2, \Sigma_1 + \mathbf{P}_2^{-1}\right). \tag{B.12}$$

Linear transformation of Gaussians,

$$\begin{aligned} \mathcal{N}\left(\mathbf{A}\mathbf{x} + \mathbf{b} \mid \mathbf{a}, \Sigma\right) &= |\mathbf{A}|^{-1} \mathcal{N}\left(\mathbf{x} \mid \mathbf{A}^{-1}\left(\mathbf{a} - \mathbf{b}\right), \mathbf{A}^{-1}\Sigma\mathbf{A}^{-T}\right) && \text{(B.13)} \\ &= |\mathbf{A}|^{-1} \mathcal{N}\left[\mathbf{x} \mid \mathbf{A}^{T}\Sigma^{-1}\left(\mathbf{a} - \mathbf{b}\right), \mathbf{A}^{T}\Sigma^{-1}\mathbf{A}\right]. && \text{(B.14)} \end{aligned}$$

$$\mathcal{N}\left[\mathbf{A}\mathbf{x} + \mathbf{b} \mid \mathbf{a}, \mathbf{P}\right] = |\mathbf{A}|^{-1} \mathcal{N}\left[\mathbf{x} \mid \mathbf{A}^{T}\left(\mathbf{a} - \mathbf{P}\mathbf{b}\right), \mathbf{A}^{T}\mathbf{P}\mathbf{A}\right]. \tag{B.15}$$

For conditionals and marginalization, we have,

$$\begin{aligned} p\left(\mathbf{x}_1, \mathbf{x}_2\right) &= \mathcal{N}\left(\begin{bmatrix}\mathbf{x}_1 \\ \mathbf{x}_2\end{bmatrix} \mid \begin{bmatrix}\hat{\mathbf{x}}_1 \\ \hat{\mathbf{x}}_2\end{bmatrix}, \begin{bmatrix}\Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22}\end{bmatrix}\right), && \text{(B.16)} \\ \Rightarrow p\left(\mathbf{x}_1 \mid \mathbf{x}_2\right) &= \mathcal{N}\left(\mathbf{x}_1 \mid \hat{\mathbf{x}}_1 + \Sigma_{12}\Sigma_{22}^{-1}\left(\mathbf{x}_2 - \hat{\mathbf{x}}_2\right), \Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21}\right), && \text{(B.17)} \\ \text{and} \quad p\left(\mathbf{x}_1\right) &= \mathcal{N}\left(\mathbf{x}_1 \mid \hat{\mathbf{x}}_1, \Sigma_{11}\right). && \text{(B.18)} \end{aligned}$$

And for cononical form,

$$\begin{aligned} p\left(\mathbf{x}_1, \mathbf{x}_2\right) &= \mathcal{N}\left[\begin{bmatrix}\mathbf{x}_1 \\ \mathbf{x}_2\end{bmatrix} \mid \begin{bmatrix}\check{\mathbf{x}}_1 \\ \check{\mathbf{x}}_2\end{bmatrix}, \begin{bmatrix}\mathbf{P}_{11} & \mathbf{P}_{12} \\ \mathbf{P}_{21} & \mathbf{P}_{22}\end{bmatrix}\right], && \text{(B.19)} \\ \Rightarrow p\left(\mathbf{x}_1 \mid \mathbf{x}_2\right) &= \mathcal{N}\left[\mathbf{x}_1 \mid \check{\mathbf{x}}_1 - \mathbf{P}_{12}\mathbf{x}_2, \mathbf{P}_{11}\right], && \text{(B.20)} \\ \text{and} \quad p\left(\mathbf{x}_1\right) &= \mathcal{N}\left[\mathbf{x}_1 \mid \check{\mathbf{x}}_1 - \mathbf{P}_{12}\mathbf{P}_{22}^{-1}\check{\mathbf{x}}_2, \mathbf{P}_{11} - \mathbf{P}_{12}\mathbf{P}_{22}^{-1}\mathbf{P}_{21}\right]. && \text{(B.21)} \end{aligned}$$

Appendix C

Deriving Messages

We start by defining the quadratic (or quadratized) cost function as,

$$\begin{aligned} c_k\left(\mathbf{x}_k, \mathbf{u}_k\right) &= \frac{1}{2}\mathbf{x}_k^T\mathbf{Q}_k\mathbf{x}_k + \frac{1}{2}\mathbf{u}_k^T\mathbf{R}_k\mathbf{u}_k + \mathbf{x}_k^T\mathbf{C}_{\mathbf{xu},k}\mathbf{u}_k \\ &- \left(\mathbf{x}_k^T\check{\mathbf{x}}_k^* + \mathbf{u}_k^T\check{\mathbf{u}}_k^*\right). \end{aligned} \tag{C.1}$$

Therefore, the reward variables at temperature η are given by,

$$\Pr\left(R_k = 1 \mid \mathbf{X}_k = \mathbf{x}_k, \mathbf{U}_k = \mathbf{u}_k\right) \propto \mathcal{N}\left[\left[\begin{array}{c} \mathbf{x}_k \\ \mathbf{u}_k \end{array}\right] \mid \eta^{-1}\left[\begin{array}{c} \check{\mathbf{x}}_k^\star \\ \check{\mathbf{u}}_k^\star \end{array}\right], \eta^{-1}\left[\begin{array}{cc} \mathbf{Q}_k & \mathbf{C}_{\mathbf{xu},k} \\ \mathbf{C}_{\mathbf{xu},k}^T & \mathbf{R}_k \end{array}\right]\right]. \tag{C.2}$$

Stacking $\mathbf{x}_k$ and $\mathbf{u}_k$ into a vector and calling it $\mathfrak{z}_k$ gives more compact equations. Thus,

$$\Pr\left(R_k = 1 \mid \mathbf{X}_k = \mathbf{x}_k, \mathbf{U}_k = \mathbf{u}_k\right) \propto \mathcal{N}\left[\mathfrak{z}_k \mid \eta^{-1}\check{\mathfrak{z}}_k^*, \eta^{-1}\left[\begin{array}{cc} \mathbf{Q}_k & \mathbf{C}_{\mathbf{xu},k} \\ \mathbf{C}_{\mathbf{xu},k}^T & \mathbf{R}_k \end{array}\right]\right] \tag{C.3}$$

In the case of linearized dynamics we have,

$$f_k\left(\mathbf{x}_{k+1} \mid \mathbf{x}_k, \mathbf{u}_k\right) = \mathcal{N}\left(\mathbf{x}_{k+1} \mid \mathbf{A}_k\mathbf{x}_k + \mathbf{B}_k\mathbf{u}_k + \mathbf{a}_k, \Sigma_{\mathbf{w}}\right). \tag{C.4}$$

where,

$$\mathbf{A}_k = \left.\frac{\partial \mathbf{f}_k\left(\mathbf{x}_k, \mathbf{u}_k, \mathbf{w}_k\right)}{\partial \mathbf{x}_k}\right|_{\mathbf{x}_k = \hat{\mathbf{x}}_k, \mathbf{u}_k = \hat{\mathbf{u}}_k, \mathbf{w}_k = 0}, \tag{C.5}$$

$$\mathbf{B}_k = \left.\frac{\partial \mathbf{f}_k\left(\mathbf{x}_k, \mathbf{u}_k, \mathbf{w}_k\right)}{\partial \mathbf{u}_k}\right|_{\mathbf{x}_k = \hat{\mathbf{x}}_k, \mathbf{u}_k = \hat{\mathbf{u}}_k, \mathbf{w}_k = 0}, \tag{C.6}$$

$$\mathbf{a}_k = \mathbf{f}_k\left(\hat{\mathbf{x}}_k, \hat{\mathbf{u}}_k, 0\right) - \left(\mathbf{A}_k\hat{\mathbf{x}}_k + \mathbf{B}_k\hat{\mathbf{u}}_k\right). \tag{C.7}$$

Again, in terms of $\mathfrak{z}_k$ one has,

$$f_k\left(\mathbf{x}_{k+1} \mid \mathbf{x}_k, \mathbf{u}_k\right) = \mathcal{N}\left(\mathbf{x}_{k+1} \mid \begin{bmatrix} \mathbf{A}_k & \mathbf{B}_k \end{bmatrix} \begin{bmatrix} \mathbf{x}_k \\ \mathbf{u}_k \end{bmatrix} + \mathbf{a}_k, \Sigma_{\mathbf{w}}\right) \tag{C.8}$$

$$= \mathcal{N}\left(\mathbf{x}_{k+1} \mid \begin{bmatrix} \mathbf{F}_k & \mathbf{B}_k \end{bmatrix} \mathfrak{z}_k + \mathbf{a}_k, \Sigma_{\mathbf{w}}\right) \tag{C.9}$$

$$= \mathcal{N}\left[\mathbf{x}_{k+1} \mid \mathbf{P}_{\mathbf{w}} \begin{bmatrix} \mathbf{F}_k & \mathbf{B}_k \end{bmatrix} \mathfrak{z}_k + \mathbf{P}_{\mathbf{w}}\mathbf{a}_k, \mathbf{P}_{\mathbf{w}}\right]. \tag{C.10}$$

The update equation for the dynamics is,

$$f_k^{(i+1)} = \frac{f_k \mathcal{T}_{\beta_k} m^{(i)}_{f_k \leftarrow \mu_{k+1}}}{\mathcal{T}_{\beta_k} m^{(i)}_{\mu_k \leftarrow f_k}}. \tag{C.11}$$

The message from μ_{k+1} to f_k is,

$$m^{(i)}_{f_k \leftarrow \mu_{k+1}} = \mathcal{N}\left(\mathbf{x}_{k+1} \mid \hat{\mathbf{x}}_{f_k \leftarrow \mu_{k+1}}, \Sigma_{f_k \leftarrow \mu_{k+1}}\right) \tag{C.12}$$

$$= \mathcal{N}\left[\mathbf{x}_{k+1} \mid \check{\mathbf{x}}_{f_k \leftarrow \mu_{k+1}}, \mathbf{P}_{f_k \leftarrow \mu_{k+1}}\right] \tag{C.13}$$

$$= \mathcal{T}^{-1}_{\alpha_{k+1}} Z_{\mu_{k+1}}\left(\alpha_{k+1}\right) \tag{C.14}$$

and the message from f_k to μ_k is,

$$m^{(i)}_{\mu_k \leftarrow f_k} = \mathcal{N}\left(\mathfrak{z}_k \mid \hat{\mathfrak{z}}_{\mu_k \leftarrow f_k}, \Sigma_{\mu_k \leftarrow f_k}\right) \tag{C.15}$$

$$= \mathcal{N}\left[\mathfrak{z}_k \mid \check{\mathfrak{z}}_{\mu_k \leftarrow f_k}, \mathbf{P}_{\mu_k \leftarrow f_k}\right]$$

$$= \mathcal{T}^{-1}_{\beta_k} Z_{f_k}\left(\beta_k\right). \tag{C.16}$$

The message from r_k to μ_k is,

$$m^{(i)}_{\mu_k \leftarrow r_k} = \mathcal{T}^{-1}_{\eta_k} e^{-\frac{c_k}{\eta}} \tag{C.17}$$

$$= \mathcal{N}\left[\mathfrak{z}_k \mid \eta^{-1}\check{\mathfrak{z}}^*_k, \eta^{-1} \begin{bmatrix} \mathbf{Q}_k & \mathbf{C}_{\mathbf{xu},k} \\ \mathbf{C}^T_{\mathbf{xu},k} & \mathbf{R}_k \end{bmatrix}\right]. \tag{C.18}$$

Starting from the numerator,

$$f_k \mathcal{T}_{\beta_k} m^{(i)}_{f_k \leftarrow \mu_{k+1}} = \mathcal{N}\left[\mathbf{x}_{k+1} \mid \mathbf{P}_{\mathbf{w}} \begin{bmatrix} \mathbf{A}_k & \mathbf{B}_k \end{bmatrix} \mathfrak{z}_k + \mathbf{P}_{\mathbf{w}}\mathbf{a}_k, \mathbf{P}_{\mathbf{w}}\right]$$
$$\times \mathcal{N}\left[\mathbf{x}_{k+1} \mid \beta_k^{-1}\check{\mathbf{x}}_{f_k \leftarrow \mu_{k+1}}, \beta_k^{-1}\mathbf{P}_{f_k \leftarrow \mu_{k+1}}\right] \tag{C.19}$$

$$= \mathcal{N}\left[\mathbf{x}_{k+1} \mid \mathbf{P}_{\mathbf{w}} \begin{bmatrix} \mathbf{A}_k & \mathbf{B}_k \end{bmatrix} \mathfrak{z}_k + \mathbf{P}_{\mathbf{w}}\mathbf{a}_k + \beta_k^{-1}\check{\mathbf{x}}_{f_k \leftarrow \mu_{k+1}}, \mathbf{P}_{\mathbf{w}} + \beta_k^{-1}\mathbf{P}_{f_k \leftarrow \mu_{k+1}}\right]$$
$$\times \mathcal{N}\left(\begin{bmatrix} \mathbf{A}_k & \mathbf{B}_k \end{bmatrix} \mathfrak{z}_k \mid \hat{\mathbf{x}}_{f_k \leftarrow \mu_{k+1}} - \mathbf{a}_k, \Sigma_{\mathbf{w}} + \beta_k \Sigma_{f_k \leftarrow \mu_{k+1}}\right). \tag{C.20}$$

For the parition function to be well-defined and finite, the following should hold. This may or may not limit the admissible range of, β_k, when it's negative.

$$\mathbf{P}_{\mathbf{w}} + \beta_k^{-1}\mathbf{P}_{f_k \leftarrow \mu_{k+1}} \succ 0, \tag{C.21}$$

and,

$$\Sigma_{\mathbf{w}} + \beta_k \Sigma_{f_k \leftarrow \mu_{k+1}} \succ 0. \tag{C.22}$$

For the denominator we have,

$$\mathcal{T}_{\beta_k} m^{(i)}_{\mu_k \leftarrow f_k} = \mathcal{N}\left(\begin{bmatrix} \mathbf{A}_k & \mathbf{B}_k \end{bmatrix} \mathfrak{z}_k \mid \hat{\mathbf{x}}_{f_k \leftarrow \mu_{k+1}} - \mathbf{a}_k, \Sigma_{\mathbf{w}} + \beta_k \Sigma_{f_k \leftarrow \mu_{k+1}}\right) \tag{C.23}$$

$$\propto \mathcal{N}\left[\mathfrak{z}_k \mid \begin{bmatrix} \mathbf{A}_k^T \\ \mathbf{B}_k^T \end{bmatrix} \left(\Sigma_{\mathbf{w}} + \beta_k \Sigma_{f_k \leftarrow \mu_{k+1}}\right)^{-1} \left(\hat{\mathbf{x}}_{f_k \leftarrow \mu_{k+1}} - \mathbf{a}_k\right) , \begin{bmatrix} \mathbf{A}_k^T \\ \mathbf{B}_k^T \end{bmatrix} \left(\Sigma_{\mathbf{w}} + \beta_k \Sigma_{f_k \leftarrow \mu_{k+1}}\right)^{-1} \begin{bmatrix} \mathbf{A}_k & \mathbf{B}_k \end{bmatrix}\right] \tag{C.24}$$

Therefore,

$$m^{(i)}_{\mu_k \leftarrow f_k} = \mathcal{N}\left[\mathfrak{z}_k \mid \check{\mathfrak{z}}_{\mu_k \leftarrow f_k}, \begin{bmatrix} \mathbf{P}_{\mathbf{xx},\mu_k \leftarrow f_k} & \mathbf{P}_{\mathbf{xu},\mu_k \leftarrow f_k} \\ \mathbf{P}_{\mathbf{ux},\mu_k \leftarrow f_k} & \mathbf{P}_{\mathbf{uu},\mu_k \leftarrow f_k} \end{bmatrix}\right], \tag{C.25}$$

where,

$$\mathbf{P}_{\mathbf{xx},\mu_k \leftarrow f_k} = \mathbf{A}_k^T \left(\beta_k^{-1}\Sigma_{\mathbf{w}} + \Sigma_{f_k \leftarrow \mu_{k+1}}\right)^{-1} \mathbf{A}_k, \tag{C.26}$$

$$\mathbf{P}_{\mathbf{xu},\mu_k \leftarrow f_k} = \mathbf{A}_k^T \left(\beta_k^{-1}\Sigma_{\mathbf{w}} + \Sigma_{f_k \leftarrow \mu_{k+1}}\right)^{-1} \mathbf{B}_k, \tag{C.27}$$

$$\mathbf{P}_{\mathbf{ux},\mu_k \leftarrow f_k} = \mathbf{B}_k^T \left(\beta_k^{-1}\Sigma_{\mathbf{w}} + \Sigma_{f_k \leftarrow \mu_{k+1}}\right)^{-1} \mathbf{A}_k, \tag{C.28}$$

$$\mathbf{P}_{\mathbf{uu},\mu_k \leftarrow f_k} = \mathbf{B}_k^T \left(\beta_k^{-1}\Sigma_{\mathbf{w}} + \Sigma_{f_k \leftarrow \mu_{k+1}}\right)^{-1} \mathbf{B}_k, \tag{C.29}$$

$$\check{\mathfrak{z}}_{\mu_k \leftarrow f_k} = \begin{bmatrix} \mathbf{A}_k^T \left(\beta_k^{-1}\Sigma_{\mathbf{w}} + \Sigma_{f_k \leftarrow \mu_{k+1}}\right)^{-1} \left(\hat{\mathbf{x}}_{f_k \leftarrow \mu_{k+1}} - \mathbf{a}_k\right) \\ \mathbf{B}_k^T \left(\beta_k^{-1}\Sigma_{\mathbf{w}} + \Sigma_{f_k \leftarrow \mu_{k+1}}\right)^{-1} \left(\hat{\mathbf{x}}_{f_k \leftarrow \mu_{k+1}} - \mathbf{a}_k\right) \end{bmatrix}. \tag{C.30}$$

The update equation for policy is,

$$\mu_k^{(i+1)} = \frac{\mu_k^{(i)} \mathcal{T}_{\alpha_k} m^{(i)}_{\mu_k \leftarrow f_k} \mathcal{T}_{\alpha_k} m^{(i)}_{\mu_k \leftarrow r_k}}{\mathcal{T}_{\alpha_k} m^{(i)}_{f_{k-1} \leftarrow \mu_k}}.$$

Suppose that from the last update of the policy we have,

$$\mu_k^{(i)} = \mathcal{N}\left(\mathbf{u}_k \mid \mathbf{K}_k^{(i)} \mathbf{x}_k + \mathbf{u}_k^{ol(i)}, \Sigma_{\mu_k^{(i)}}\right). \tag{C.31}$$

Before deriving the numinator, let us rearrange $\mu_k^{(i)}$ in terms of $\mathfrak{z}_k$,

$$\mathcal{N}\left(\mathbf{u}_k \mid \mathbf{K}_k^{(i)}\mathbf{x}_k + \mathbf{u}_k^{ol(i)}, \Sigma_{\mu_k^{(i)}}\right) = \mathcal{N}\left(\begin{bmatrix}-\mathbf{K}_k^{(i)} & I\end{bmatrix}\mathfrak{z}_k \mid \mathbf{u}_k^{ol(i)}, \Sigma_{\mu_k^{(i)}}\right) \tag{C.32}$$

$$= \mathcal{N}\left[\begin{bmatrix}-\mathbf{K}_k^{(i)} & I\end{bmatrix}\mathfrak{z}_k \mid \Sigma^{-1}_{\mu_k^{(i)}}\mathbf{u}_k^{ol(i)}, \Sigma^{-1}_{\mu_k^{(i)}}\right] \tag{C.33}$$

$$\propto \mathcal{N}\left[\mathfrak{z}_k \mid \begin{bmatrix}-\mathbf{K}_k^{(i)} & I\end{bmatrix}^T \Sigma^{-1}_{\mu_k^{(i)}}\mathbf{u}_k^{ol(i)}, \begin{bmatrix}-\mathbf{K}_k^{(i)} & I\end{bmatrix}^T \Sigma^{-1}_{\mu_k^{(i)}}\begin{bmatrix}-\mathbf{K}_k^{(i)} & I\end{bmatrix}\right] \tag{C.34}$$

$$= \mathcal{N}\left[\mathfrak{z}_k \mid \begin{bmatrix}-\mathbf{K}_k^{(i)T}\Sigma^{-1}_{\mu_k^{(i)}}\mathbf{u}_k^{ol(i)} \\ I^T\Sigma^{-1}_{\mu_k^{(i)}}\mathbf{u}_k^{ol(i)}\end{bmatrix}, \begin{bmatrix}\mathbf{K}_k^{(i)T}\Sigma^{-1}_{\mu_k^{(i)}}\mathbf{K}_k^{(i)} & -\mathbf{K}_k^{(i)T}\Sigma^{-1}_{\mu_k^{(i)}} \\ -\Sigma^{-1}_{\mu_k^{(i)}}\mathbf{K}_k^{(i)} & \Sigma^{-1}_{\mu_k^{(i)}}\end{bmatrix}\right]. \tag{C.35}$$

Now, starting from the numinator,

$$\mu_k^{(i)}\mathcal{T}_{\alpha_k}m^{(i)}_{\mu_k\leftarrow f_k}\mathcal{T}_{\alpha_k}m^{(i)}_{\mu_k\leftarrow r_k} \quad = \quad \mu_k^{(i)}e^{-\frac{c_k-\beta_k\log Z_{f_k}(\beta_k)}{\alpha_k}}, \tag{C.36}$$

we have

$$\begin{aligned}\mu_k^{(i)}\mathcal{T}_{\alpha_k}m^{(i)}_{\mu_k\leftarrow f_k}\mathcal{T}_{\alpha_k}m^{(i)}_{\mu_k\leftarrow r_k} &= \underbrace{\mathcal{N}\left(\mathbf{u}_k \mid \mathbf{K}_k^{(i)}\mathbf{x}_k + \mathbf{u}_k^{ol(i)}, \Sigma_{\mu_k^{(i)}}\right)}_{(C.37.I)} \\ &\times \underbrace{\mathcal{N}\left[\mathfrak{z}_k \mid \alpha_k^{-1}\check{\mathfrak{z}}_{\mu_k\leftarrow f_k}, \begin{bmatrix}\alpha_k^{-1}\mathbf{P}_{\mathbf{xx},\mu_k\leftarrow f_k} & \alpha_k^{-1}\mathbf{P}_{\mathbf{xu},\mu_k\leftarrow f_k} \\ \alpha_k^{-1}\mathbf{P}_{\mathbf{ux},\mu_k\leftarrow f_k} & \alpha_k^{-1}\mathbf{P}_{\mathbf{uu},\mu_k\leftarrow f_k}\end{bmatrix}\right]}_{(C.37.II)} \\ &\times \underbrace{\mathcal{N}\left[\mathfrak{z}_k \mid \alpha_k^{-1}\check{\mathfrak{z}}_{\mu_k\leftarrow f_k}, \begin{bmatrix}\alpha_k^{-1}\mathbf{P}_{\mathbf{xx},\mu_k\leftarrow f_k} & \alpha_k^{-1}\mathbf{P}_{\mathbf{xu},\mu_k\leftarrow f_k} \\ \alpha_k^{-1}\mathbf{P}_{\mathbf{ux},\mu_k\leftarrow f_k} & \alpha_k^{-1}\mathbf{P}_{\mathbf{uu},\mu_k\leftarrow f_k}\end{bmatrix}\right]}_{(C.37.III)}.\end{aligned} \tag{C.37}$$

First, we multiply the two first of the product above,

$$(C.37.I)\times(C.37.II) = \tag{C.38}$$

$$\times \mathcal{N}\left[\mathfrak{z}_k \mid \begin{bmatrix}-\mathbf{K}_k^{(i)T}\Sigma^{-1}_{\mu_k^{(i)}}\mathbf{u}_k^{ol(i)} \\ I^T\Sigma^{-1}_{\mu_k^{(i)}}\mathbf{u}_k^{ol(i)}\end{bmatrix}, \begin{bmatrix}\mathbf{K}_k^{(i)T}\Sigma^{-1}_{\mu_k^{(i)}}\mathbf{K}_k^{(i)} & -\mathbf{K}_k^{(i)T}\Sigma^{-1}_{\mu_k^{(i)}} \\ -\Sigma^{-1}_{\mu_k^{(i)}}\mathbf{K}_k^{(i)} & \Sigma^{-1}_{\mu_k^{(i)}}\end{bmatrix}\right] \tag{C.39}$$

$$\times \mathcal{N}\left[\mathfrak{z}_k \mid \alpha_k^{-1}\check{\mathfrak{z}}_{\mu_k\leftarrow f_k}, \begin{bmatrix}\alpha_k^{-1}\mathbf{P}_{\mathbf{xx},\mu_k\leftarrow f_k} & \alpha_k^{-1}\mathbf{P}_{\mathbf{xu},\mu_k\leftarrow f_k} \\ \alpha_k^{-1}\mathbf{P}_{\mathbf{ux},\mu_k\leftarrow f_k} & \alpha_k^{-1}\mathbf{P}_{\mathbf{uu},\mu_k\leftarrow f_k}\end{bmatrix}\right]. \tag{C.40}$$

Therefore, we have,

$$(C.37.I) \times (C.37.II) \propto \tag{C.41}$$

$$\mathcal{N}\left[\mathfrak{z}_k \mid \begin{bmatrix} -\mathbf{K}_k^{(i)T}\Sigma^{-1}_{\mu_k^{(i)}}\mathbf{u}_k^{ol(i)} \\ I^T\Sigma^{-1}_{\mu_k^{(i)}}\mathbf{u}_k^{ol(i)} \end{bmatrix} + \alpha_k^{-1}\check{\mathfrak{z}}_{\mu_k\leftarrow f_k}, \tag{C.42}$$

$$, \begin{bmatrix} \alpha_k^{-1}\mathbf{P}_{\mathbf{xx},\mu_k\leftarrow f_k} & \alpha_k^{-1}\mathbf{P}_{\mathbf{xu},\mu_k\leftarrow f_k} \\ \alpha_k^{-1}\mathbf{P}_{\mathbf{ux},\mu_k\leftarrow f_k} & \alpha_k^{-1}\mathbf{P}_{\mathbf{uu},\mu_k\leftarrow f_k} \end{bmatrix} + \begin{bmatrix} \mathbf{K}_k^{(i)T}\Sigma^{-1}_{\mu_k^{(i)}}\mathbf{K}_k^{(i)} & -\mathbf{K}_k^{(i)T}\Sigma^{-1}_{\mu_k^{(i)}} \\ -\Sigma^{-1}_{\mu_k^{(i)}}\mathbf{K}_k^{(i)} & \Sigma^{-1}_{\mu_k^{(i)}} \end{bmatrix}.\Bigg] \tag{C.43}$$

Continuing with the third term in the product,

$$\mu_k^{(i)}\mathcal{T}_{\alpha_k}m^{(i)}_{\mu_k\leftarrow f_k}\mathcal{T}_{\alpha_k}m^{(i)}_{\mu_k\leftarrow r_k} \propto (C.37.I)\times(C.37.II)\times\mathcal{N}\left[\mathfrak{z}_k \mid \alpha_k^{-1}\check{\mathfrak{z}}_k^*, \alpha_k^{-1}\begin{bmatrix} \mathbf{Q}_k & \mathbf{C}_{\mathbf{xu},k} \\ \mathbf{C}_{\mathbf{ux},k} & \mathbf{R}_k \end{bmatrix}\right], \tag{C.44}$$

which after some simlifications becomes,

$$\mu_k^{(i)}\mathcal{T}_{\alpha_k}m^{(i)}_{\mu_k\leftarrow f_k}\mathcal{T}_{\alpha_k}m^{(i)}_{\mu_k\leftarrow r_k} = \tag{C.45}$$

$$\mathcal{N}\left[\mathfrak{z}_k \mid \begin{bmatrix} -\mathbf{K}_k^{(i)T}\Sigma^{-1}_{\mu_k^{(i)}}\mathbf{u}_k^{ol(i)} \\ \Sigma^{-1}_{\mu_k^{(i)}}\mathbf{u}_k^{ol(i)} \end{bmatrix} + \alpha_k^{-1}\check{\mathfrak{z}}_{\mu_k\leftarrow f_k} + \alpha_k^{-1}\check{\mathfrak{z}}_k^*, \tag{C.46}$$

$$\begin{bmatrix} \alpha_k^{-1}\mathbf{P}_{\mathbf{xx},\mu_k\leftarrow f_k} & \alpha_k^{-1}\mathbf{P}_{\mathbf{xu},\mu_k\leftarrow f_k} \\ \alpha_k^{-1}\mathbf{P}_{\mathbf{ux},\mu_k\leftarrow f_k} & \alpha_k^{-1}\mathbf{P}_{\mathbf{uu},\mu_k\leftarrow f_k} \end{bmatrix} + \begin{bmatrix} \mathbf{K}_k^{(i)T}\Sigma^{-1}_{\mu_k^{(i)}}\mathbf{K}_k^{(i)} & -\mathbf{K}_k^{(i)T}\Sigma^{-1}_{\mu_k^{(i)}} \\ -\Sigma^{-1}_{\mu_k^{(i)}}\mathbf{K}_k^{(i)} & \Sigma^{-1}_{\mu_k^{(i)}} \end{bmatrix} + \begin{bmatrix} \alpha_k^{-1}\mathbf{Q}_k & \alpha_k^{-1}\mathbf{C}_{\mathbf{xu},k} \\ \alpha_k^{-1}\mathbf{C}_{\mathbf{ux},k} & \alpha_k^{-1}\mathbf{R}_k \end{bmatrix}\Bigg]. \tag{C.47}$$

Therefore,

$$\begin{aligned}\mathbf{P}^{(i+1)}_{f_{k-1}\leftarrow\mu_k} = &\left(\mathbf{Q}_k + \alpha_k\mathbf{K}_k^{(i)T}\Sigma^{-1}_{\mu_k^{(i)}}\mathbf{K}_k^{(i)} + \mathbf{P}_{\mathbf{xx},\mu_k\leftarrow f_k}\right) \\ &+ \left(\mathbf{C}_{\mathbf{xu},k} + \mathbf{P}_{\mathbf{xu},\mu_k\leftarrow f_k} - \alpha_k\mathbf{K}_k^{(i)T}\Sigma^{-1}_{\mu_k^{(i)}}\right)\mathbf{K}_k^{(i+1)}\end{aligned} \tag{C.48}$$

$$\check{\mathbf{x}}^{(i+1)}_{f_{k-1}\leftarrow\mu_k} \tag{C.49}$$

$$= \check{\mathbf{x}}_{\mu_k\leftarrow f_k} + \check{\mathbf{x}}_k^* - \alpha_k\mathbf{K}_k^{(i)T}\Sigma^{-1}_{\mu_k^{(i)}}\mathbf{u}_k^{ol(i)} + \mathbf{K}_k^{(i+1)T}\left(\alpha_k\Sigma^{-1}_{\mu_k^{(i)}}\mathbf{u}_k^{ol(i)} + \check{\mathbf{u}}_{\mu_k\leftarrow f_k} + \check{\mathbf{u}}_k^*\right) \tag{C.50}$$

$$= \check{\mathbf{x}}_{\mu_k\leftarrow f_k} + \check{\mathbf{x}}_k^* + \mathbf{K}_k^{(i+1)T}\left(\check{\mathbf{u}}_{\mu_k\leftarrow f_k} + \check{\mathbf{u}}_k^*\right) + \alpha_k\left(\mathbf{K}_k^{(i+1)T} - \mathbf{K}_k^{(i)T}\right)\Sigma^{-1}_{\mu_k^{(i)}}\mathbf{u}_k^{ol(i)}. \tag{C.51}$$

Therefore, the updated policies becomes,

$$\mathcal{N}\left(\mathbf{u}_k \mid \mathbf{K}_k^{(i+1)}\mathbf{x}_k + \mathbf{u}_k^{ol(i+1)}, \Sigma_{\mu_k^{(i+1)}}\right), \tag{C.52}$$

where,

$$\Sigma_{\mu_k^{(i+1)}} = \left(\alpha_k^{-1}\mathbf{R}_k + \Sigma^{-1}_{\mu_k^{(i)}} + \alpha_k^{-1}\mathbf{P}_{\mathbf{uu},\mu_k \leftarrow f_k}\right)^{-1} \tag{C.53}$$

$$\mathbf{K}_k^{(i+1)} = \Sigma_{\mu_k^{(i+1)}}\left(\Sigma^{-1}_{\mu_k^{(i)}}\mathbf{K}_k^{(i)} - \alpha_k^{-1}\mathbf{P}_{\mathbf{ux},\mu_k \leftarrow f_k} - \alpha_k^{-1}\mathbf{C}_{\mathbf{ux},k}\right) \tag{C.54}$$

$$\mathbf{u}_k^{ol(i+1)} = \Sigma_{\mu_k^{(i+1)}}\left(\Sigma^{-1}_{\mu_k^{(i)}}\mathbf{u}_k^{ol(i)} + \alpha_k^{-1}\check{\mathbf{u}}_{\mu_k \leftarrow f_k} + \alpha_k^{-1}\check{\mathbf{u}}_k^*\right). \tag{C.55}$$

Appendix D

Kalman Filtering

D.1 The standard Kalman filter.

KF, presented by Kalman in his seminal work [7], is probably the most significant result in the theory of filtering. Here we derived it from a Bayesian point of view. First, let's assume a linear dynamics and measurement equation, as in,

$$\mathbf{x}_{k+1} = \mathbf{A}_k \mathbf{x}_k + \mathbf{B}_k \mathbf{u}_k + \mathbf{w}_k. \tag{D.1}$$

$$\mathbf{y}_k = \mathbf{H}_k \mathbf{x}_k + \mathbf{v}_k. \tag{D.2}$$

And we further assume that noises are normal and i.d.d. , with distributions $\mathbf{w}_k \sim \mathcal{N}(\mathbf{w}_k \mid 0, \Sigma_{\mathbf{w}})$ and $\mathbf{v}_k \sim \mathcal{N}(\mathbf{v}_k \mid 0, \Sigma_{\mathbf{v}})$. To find the recursive solution, we assume that the belief at time $b_{k-1}(\mathbf{x}_{k-1})$ is available,

$$b_{k-1}(\mathbf{x}_{k-1}) = \mathcal{N}(\mathbf{x}_{k-1} \mid \hat{\mathbf{x}}_{k-1}, \Sigma_{\mathbf{x-1}}), \tag{D.3}$$

and find the belief at $b_k(\mathbf{x}_k)$, given $b_{k-1}(\mathbf{x}_{k-1})$, the latest measurement, $\underline{\mathbf{y}}_k$ and the previous control signal, $\underline{\mathbf{u}}_{k-1}$. The linear dynamics of the plant results in the following conditional probability,

$$p(\mathbf{x}_k \mid \mathbf{x}_{k-1}, \mathbf{u}_{k-1}) = \mathcal{N}(\mathbf{x}_k \mid \mathbf{A}_{k-1}\mathbf{x}_{k-1} + \mathbf{B}_{k-1}\mathbf{u}_{k-1}, \Sigma_{\mathbf{w}}). \tag{D.4}$$

We can now perform the prediction step:

$$\begin{aligned} b_k^-(\mathbf{x}_k) &= \int_{\mathbf{x}_{k-1}} \mathcal{N}\left(\mathbf{x}_k \mid \mathbf{A}_{k-1}\mathbf{x}_{k-1} + \mathbf{B}_{k-1}\underline{\mathbf{u}}_{k-1}, \Sigma_{\mathbf{w}}\right) \mathcal{N}(\mathbf{x}_{k-1} \mid \hat{\mathbf{x}}_{k-1}, \Sigma_{\mathbf{x-1}}) && \text{(D.5)} \\ &= \mathcal{N}\left(\mathbf{x}_k \mid \mathbf{A}_{k-1}\hat{\mathbf{x}}_{k-1} + \mathbf{B}_{k-1}\underline{\mathbf{u}}_{k-1}, \mathbf{A}_{k-1}\Sigma_{\mathbf{x-1}}\mathbf{A}_{k-1}^T + \Sigma_{\mathbf{w}}\right) && \text{(D.6)} \\ &= \mathcal{N}\left(\mathbf{x}_k \mid \hat{\mathbf{x}}_k^-, \Sigma_{\mathbf{x}_k}^-\right) && \text{(D.7)} \end{aligned}$$

The linear measurement equation results in the following conditional,

$$p(\mathbf{y}_k \mid \mathbf{x}_k) = \mathcal{N}(\mathbf{y}_k \mid \mathbf{H}_k\mathbf{x}_k, \Sigma_{\mathbf{v}}). \tag{D.8}$$

We can predict the measurement given the predicted state,

$$\begin{aligned} p\left(\mathbf{y}_k \mid \underline{\mathbf{u}}_{k-1}, \mathscr{I}_{k-1}\right) &= \int_{\mathbf{x}_k} p\left(\mathbf{y}_k \mid \mathbf{x}_k\right) b_k^-\left(\mathbf{x}_k\right) && \text{(D.9)} \\ &= \mathcal{N}\left(\mathbf{y}_k \mid \mathbf{H}_k \hat{\mathbf{x}}_k, \mathbf{H}_k \Sigma_{\mathbf{w}} \mathbf{H}_k^T + \Sigma_{\mathbf{v}}\right) && \text{(D.10)} \\ &= \mathcal{N}\left(\mathbf{y}_k \mid \hat{\mathbf{y}}_k^-, \mathbf{S}_k\right) && \text{(D.11)} \end{aligned}$$

We can now find the joint distribution of $\mathbf{y}_k, \mathbf{x}_k$ given all previous measurements and controls,

$$p\left(\mathbf{y}_k, \mathbf{x}_k \mid \underline{\mathbf{u}}_{k-1}, \mathscr{I}_{k-1}\right) = \mathcal{N}\left(\left[\begin{array}{c} \mathbf{x}_k \\ \mathbf{y}_k \end{array}\right] \middle| \left[\begin{array}{c} \hat{\mathbf{x}}_k^- \\ \hat{\mathbf{y}}_k^- \end{array}\right], \left[\begin{array}{cc} \Sigma_{\mathbf{x}_k}^- & \Sigma_{\mathbf{x}_k}^- \mathbf{H}_k^T \\ \mathbf{H}_k \Sigma_{\mathbf{x}_k}^- & \mathbf{S}_k \end{array}\right]\right). \tag{D.12}$$

From Appendix B on properties of joint Gaussian random variables, we can find $p\left(\mathbf{x}_k \mid \mathbf{y}_k, \mathscr{I}_{k-1}, \mathbf{u}_{k-1}\right)$,

$$p\left(\mathbf{x}_k \mid \mathbf{y}_k, \underline{\mathbf{u}}_{k-1}, \mathscr{I}_{k-1}\right) = \mathcal{N}\left(\mathbf{x}_k \mid \hat{\mathbf{x}}_k^- + \mathbf{K}_k\left(\mathbf{y}_k - \hat{\mathbf{y}}_k^-\right), \Sigma_{\mathbf{x}_k}\right), \tag{D.13}$$

where,

$$\begin{aligned} \Sigma_{\mathbf{x}_k} &= \Sigma_{\mathbf{x}_k}^- - \mathbf{K}_k \mathbf{S}_k \mathbf{K}_k^T, && \text{(D.14)} \\ \mathbf{K}_k &= \Sigma_{\mathbf{x}_k} \mathbf{H}_k^T \mathbf{S}_k^{-1}, && \text{(D.15)} \\ \hat{\mathbf{x}}_k &= \hat{\mathbf{x}}_k^- + \mathbf{K}_k\left(\mathbf{y}_k - \hat{\mathbf{y}}_k^-\right). && \text{(D.16)} \end{aligned}$$

Therefore, we only need to replace the variable $\mathbf{y}_k$, with the value of the latest measurement, $\underline{\mathbf{y}}_k$, to find the belief, i.e.,

$$\begin{aligned} b_k\left(\mathbf{x}_k\right) &= \left(\mathbf{x}_k \mid \mathscr{I}_k\right) && \text{(D.17)} \\ &= p\left(\mathbf{x}_k \mid \underline{\mathbf{y}}_k, \underline{\mathbf{u}}_{k-1}, \mathscr{I}_{k-1}\right) && \text{(D.18)} \\ &= \mathcal{N}\left(\mathbf{x}_k \mid \hat{\mathbf{x}}_k^- + \mathbf{K}_k\left(\underline{\mathbf{y}}_k - \hat{\mathbf{y}}_k^-\right), \Sigma_{\mathbf{x}_k}\right). && \text{(D.19)} \end{aligned}$$

This completes the derivation of KF.

D.2 The Extended Kalman Filter

We assumed a linear dynamics to derive KF. However, for many practical problems the dynamics is nonlinear and, therefore, Kalman filter cannot be readily applied. We could simply linearize the plant and apply the Kalman filter, as a local approximation of the complete Bayes recursion. If the region where the minimization is accurate is large enough this approximation is useful. Another choice is EKF which uses the two following approximations:

$$\begin{aligned} E\left[f\left(x\right)\right] &= f\left(E\left[x\right]\right), && \text{(D.20)} \\ cov\left(f\left(x\right)\right) &= f'\left(x\right) cov(x) f'\left(x\right)^T. && \text{(D.21)} \end{aligned}$$

That is to say, the covariance is approximated as if the plant was linearized, but the mean is calculated using the nonlinear dynamics. With these assumptions, the prediction step becomes,

$$b_k^- (\mathbf{x}_k) = \mathcal{N}\left(\mathbf{x}_k \mid \hat{\mathbf{x}}_k^-, \Sigma_{\mathbf{x}_k}^-\right), \tag{D.22}$$

where,

$$\begin{aligned}
\hat{\mathbf{x}}_k^- &= \mathbf{f}(\hat{\mathbf{x}}_{k-1}, \underline{\mathbf{u}}_{k-1}, 0, k), && \text{(D.23)}\\
\Sigma_{\mathbf{x}_k}^- &= \mathbf{F}_{k-1}\Sigma_{\mathbf{x}-\mathbf{1}}\mathbf{F}_{k-1}^T + \Sigma_{\mathbf{w}}, && \text{(D.24)}\\
\mathbf{F}_{k-1} &= \left.\frac{\partial \mathbf{f}(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}, \mathbf{w}_{k-1}, k)}{\partial \mathbf{x}_{k-1}}\right|_{\mathbf{x}_{k-1}=\hat{\mathbf{x}}_{k-1}, \mathbf{u}_{k-1}=\underline{\mathbf{u}}_{k-1}, \mathbf{w}_{k-1}=0}. && \text{(D.25)}
\end{aligned}$$

The update step becomes,

$$b_k (\mathbf{x}_k) = \mathcal{N}\left(\mathbf{x}_k \mid \hat{\mathbf{x}}_k^- + \mathbf{K}_k\left(\underline{\mathbf{y}}_k - \hat{\mathbf{y}}_k^-\right), \Sigma_{\mathbf{x}_k}\right), \tag{D.26}$$

where,

$$\begin{aligned}
\hat{\mathbf{y}}_k^- &= \mathbf{g}\left(\hat{\mathbf{x}}_k^-, 0, k\right), && \text{(D.27)}\\
\Sigma_{\mathbf{x}_k} &= \Sigma_{\mathbf{x}_k}^- - \mathbf{K}_k\mathbf{S}_k\mathbf{K}_k^T, && \text{(D.28)}\\
\mathbf{K}_k &= \Sigma_{\mathbf{x}_k}\mathbf{G}_k^T\mathbf{S}_k^{-1}, && \text{(D.29)}\\
\mathbf{G}_k &= \left.\frac{\partial \mathbf{g}(\mathbf{x}_k, \mathbf{v}_k, k)}{\partial \mathbf{x}_k}\right|_{\mathbf{x}_{k-1}=\hat{\mathbf{x}}_k, \mathbf{v}_k=0}. && \text{(D.30)}
\end{aligned}$$

Bibliography

[1] H. Sussmann and J. Willems, "300 years of optimal control: From the brachystochrone to the maximum principle," *IEEE Control Systems Magazine*, vol. 17, no. 3, pp. 32–44, 1997. (Cited on page 1.)

[2] B. V. G. R. . M. E. Pontryagin, L., *The Mathematical Theory of Optimal Processes*. New York City, NY: Pergamon Press, 1962. (Cited on page 1.)

[3] R. Bellman, *Dynamic Programming*. Princeton, NJ: Princeton University Press, 1957. (Cited on pages 1 and 14.)

[4] R. Bellman, "On the theory of dynamic programming," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 38, no. 8, pp. 716–802, 1952. (Cited on pages 1 and 14.)

[5] R. E. Kalman, "On the general theory of control systems," *IRE Transactions on Automatic Control*, vol. 4, no. 3, pp. 110–111, 1959. (Cited on page 1.)

[6] R. E. Kalman, "Contributions to the theory of optimal control," *Boletin Sociedad Matemática Mexicana*, vol. 5, no. 2, pp. 102–119, 1960. (Cited on page 16.)

[7] R. E. Kalman and R. S. Bucy, "New results in linear filtering and prediction theory," *Journal of Basic Engineering*, vol. 83, no. 3, pp. 95–108, 1961. (Cited on pages 1 and 102.)

[8] P. H. Zipkin, *Foundations of inventory management*, vol. 2. New York City, NY: McGraw-Hill, 2000. (Cited on page 2.)

[9] J. Stein, *Stochastic Optimal Control and the U.S. Financial Debt Crisis*. SpringerLink : Bücher, Berlin, Germany: Springer, 2012. (Cited on page 2.)

[10] W. Powell, *Approximate Dynamic Programming: Solving the Curses of Dimensionality*. Wiley Series in Probability and Statistics, Hoboken, NJ: Wiley, 2011. (Cited on pages 2, 3, and 4.)

[11] M. Toussaint, "Probabilistic inference as a model of planned behavior," *Künstliche Intelligenz*, vol. 3, no. 9, pp. 23–29, 2009. (Cited on page 3.)

[12] E. Todorov, "General duality between optimal control and estimation," in *Proc. 47th IEEE Conference on Decision and Control*, pp. 4286–4292, IEEE, 2008. (Cited on pages 4 and 6.)

[13] H. J. Kappen, "Linear theory for control of nonlinear stochastic systems," *Physical Review Letters*, vol. 95, pp. 200–201, Nov 2005. (Cited on pages 4 and 5.)

[14] E. Todorov, "Efficient computation of optimal actions," *Proceedings of the National Academy of Sciences*, vol. 106, no. 28, pp. 11478–11483, 2009. (Cited on page 4.)

[15] M. Toussaint and A. Storkey, "Probabilistic inference for solving discrete and continuous state markov decision processes," in *Proc. 23rd International Conference on Machine Learning*, pp. 945–952, ACM, 2006. (Cited on page 3.)

[16] K. Rawlik, M. Toussaint, and S. Vijayakumar, "On stochastic optimal control and reinforcement learning by approximate inference," in *Proc. International Conference on Robotics Science and Systems*, MIT Press, 2012. (Cited on pages 3, 4, 5, 6, 7, 18, 19, 20, 21, 34, 40, 42, 43, 48, 50, 54, 56, 88, and 91.)

[17] M. Hoffman, N. de Freitas, A. Doucet, and J. Peters, "An expectation-maximization algorithm for continuous markov decision processes with arbitrary rewards," in *Proc. 12th International Conference on Artificial Intelligence and Statistics*, (Clearwater Beach, FL), pp. 232–239, 2009. (Cited on pages 4, 6, and 56.)

[18] C. Andrieu, A. Doucet, S. Singh, and V. TADIC, "Particle methods for change detection, system identification, and control," *Proceedings of the IEEE*, vol. 92, no. 3, pp. 423–438, 2004. (Not cited.)

[19] M. Hoffman, H. Kueck, N. de Freitas, and A. Doucet, "New inference strategies for solving markov decision processes using reversible jump MCMC," in *Proc. 25th Conference on Uncertainty in Artificial Intelligence*, (Montreal, Canada), pp. 223–231, AUAI Press, 2009. (Cited on page 44.)

[20] D. Barber and T. Furmston, "Solving deterministic policy (PO) MDPs using expectation-maximisation and antifreeze," in *Proc. European Conference on Machine Learning*, vol. 50, (Bled, Slovenia), p. 64, 2009. (Cited on page 4.)

[21] T. Furmston and D. Barber, "Variational methods for reinforcement learning," in *Proc. 13th International Conference on Artificial Intelligence and Statistics*, vol. 9, (Sardinia, Italy), pp. 241–248, 2010. (Not cited.)

[22] H. Kappen, V. Gómez, and M. Opper, "Optimal control as a graphical model inference problem," *Machine Learning*, pp. 1–24, 2009. (Cited on pages 4, 6, and 53.)

[23] Q. Liu and A. Ihler, "Belief propagation for structured decision making," in *Proc. Uncertainty in Artificial Intelligence*, (Catalina Island, CA), 2012. (Cited on pages 19, 40, and 56.)

[24] M. Toussaint, A. Storkey, and S. Harmeling, *Expectation-Maximization methods for solving (PO) MDPs and optimal control problems*, ch. 18, pp. 388–413. Cambridge University Press, bayesian time series models ed., 2011. (Cited on pages 3, 4, and 56.)

[25] E. Theodorou, J. Buchli, and S. Schaal, "Reinforcement learning of motor skills in high dimensions: A path integral approach," in *International Conference on Robotics and Automation*, (Anchorage, AK), pp. 2397–2403, IEEE, 2010. (Cited on page 3.)

[26] J. Kober and J. Peters, "Policy search for motor primitives in robotics," *Machine Learning*, vol. 84, no. 1-2, pp. 171–203, 2011. (Not cited.)

[27] M. Toussaint and C. Goerick, "Probabilistic inference for structured planning in robotics," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3068–3073, IEEE, 2007. (Cited on pages 19 and 48.)

[28] M. Toussaint, "Robot trajectory optimization using approximate inference," in *Proc. 26th Annual International Conference on Machine Learning*, pp. 1049–1056, ACM, 2009. (Cited on pages 4, 18, 19, 34, 48, and 49.)

[29] D. Verma and R. P. Rao, "Planning and acting in uncertain environments using probabilistic inference," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2382–2387, IEEE, 2006. (Not cited.)

[30] M. Toussaint, N. Plath, T. Lang, and N. Jetchev, "Integrated motor control, planning, grasping and high-level reasoning in a blocks world using probabilistic inference," in *Proc. IEEE International Conference on Robotics and Automation*, pp. 385 –391, may 2010. (Cited on pages 3 and 5.)

[31] R. Howard, *Dynamic programming and Markov processes*. Cambridge, MA: Technology Press of Massachusetts Institute of Technology, 1960. (Cited on page 3.)

[32] C. Derman, *Finite State Markovian Decision Processes*. Mathematics in Science and Engineering, Waltham, MA: Academic Press, 1970. (Not cited.)

[33] D. Heyman and M. Sobel, *Stochastic Models in Operations Research: Stochastic optimization*. McGraw-Hill Series in Quantitative Methods for Management, New York City, NY: McGraw-Hill, 1984. (Not cited.)

[34] R. Bellman, "A markovian decision process," *Indiana University Math Journal*, vol. 6, pp. 679–684, 1957. (Cited on page 3.)

[35] M. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley Series in Probability and Statistics, Hoboken, NJ: Wiley-Interscience, 2005. (Cited on page 3.)

[36] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *Journal of Artificial Intelligence Research*, vol. 4, no. 1, pp. 237–285, 1996. (Cited on page 4.)

[37] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, vol. 1. Cambridge, England: Cambridge University Press, 1998. (Cited on page 4.)

[38] P. J. Werbos, *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. PhD thesis, Harvard University, 1974. (Cited on page 4.)

[39] P. J. Werbos, "Approximate dynamic programming for real-time control and neural modeling," *Handbook of intelligent control: Neural, fuzzy, and adaptive approaches*, vol. 15, pp. 493–525, 1992. (Cited on page 4.)

[40] D. Bertsekas and C. White, "Dynamic programming and stochastic control," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 7, no. 10, pp. 758–759, 1977. (Cited on page 4.)

[41] J. Tsitsiklis, "Asynchronous stochastic approximation and q-learning," *Machine Learning*, vol. 16, no. 3, pp. 185–202, 1994. (Cited on page 4.)

[42] D. Bertsekas and J. Tsitsiklis, *Neuro-Dynamic Programming*. Athena Scientific optimization and computation series, Nashua, NH: Athena Scientific, 1996. (Cited on pages 4 and 17.)

[43] D. Bertsekas, *Dynamic Programming and Optimal Control*, vol. I & II of *Athena Scientific optimization and computation series*. Nashua, NH: Athena Scientific, third ed., 2005. (Cited on pages 4, 11, 13, 14, and 21.)

[44] J. Si, A. Barto, and W. Powell, *Handbook of Learning and Approximate Dynamic Programming*. IEEE Press Series on Computational Intelligence, Hoboken, NJ: Wiley-IEEE Press, 2004. (Cited on page 4.)

[45] D. Bertsekas, *Dynamic Programming and Optimal Control*, vol. I of *Athena Scientific optimization and computation series*. Nashua, NH: Athena Scientific, fourth ed., 2012. (Cited on pages 4 and 17.)

[46] C. D. Charalambous, F. Rezaei, and A. Kyprianou, "Relations between information theory, robustness,and statistical mechanics of stochastic systems," in *Proc. 43rd IEEE Conference on Decision and Control*, vol. 4, pp. 3479–3484, IEEE, 2004. (Cited on page 4.)

[47] D. Wolpert, "Information theory–the bridge connecting bounded rational game theory and statistical physics," *Complex Engineered Systems*, pp. 262–290, 2006. (Cited on pages 6, 19, and 54.)

[48] M. Mezard and A. Montanari, *Information, Physics, and Computation*. Oxford, UK: Oxford University Press, 2009. (Cited on pages 4 and 93.)

[49] *Statistical Physics of Inference And Control*, Sept. 2012. (Cited on page 4.)

[50] K. Dvijotham and E. Todorov, *Reinforcement Learning and Approximate Dynamic Programming for Feedback Control*, ch. Linearly Solvable Optimal Control, pp. 119–141. Hoboken, NJ: Wiley-IEEE Press, 2012. (Cited on pages 4, 5, and 53.)

[51] E. Todorov, "Linearly-solvable markov decision problems," *Advances in Neural Information Processing Systems*, vol. 19, p. 1369, 2007. (Cited on pages 4, 5, and 20.)

[52] M. Toussaint and C. Goerick, "A bayesian view on motor control and planning," *From Motor Learning to Interaction Learning in Robots*, pp. 227–252, 2010. (Cited on page 5.)

[53] K. Friston, "What is optimal about motor control?," *Neuron*, vol. 72, no. 3, pp. 488–498, 2011. (Not cited.)

[54] S. Schaal and N. Schweighofer, "Computational motor control in humans and robots," *Current Opinion in Neurobiology*, vol. 15, no. 6, pp. 675 – 682, 2005. <ce:title>Motor sytems / Neurobiology of behaviour</ce:title>. (Not cited.)

[55] S. Schaal and C. Atkeson, "Learning control in robotics," *Robotics & Automation Magazine, IEEE*, vol. 17, no. 2, pp. 20–29, 2010. (Cited on page 5.)

[56] M. Gheshlaghi Azar, V. Gomez, and H. J. Kappen, "Dynamic policy programming," *Journal of Machine Learning Research*, vol. 13, pp. 3207–3245, 2012. (Cited on pages 6, 54, and 90.)

[57] P. A. Ortega and D. A. Braun, "Thermodynamics as a theory of decision-making with information-processing costs," *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Science*, vol. 469, no. 2153, 2013. (Cited on pages 6, 34, 54, and 55.)

[58] P. A. Ortega and D. A. Braun, "A minimum relative entropy principle for learning and acting," *Journal of Artificial Intelligence Research*, vol. 38, no. 2153, pp. 475–511, 2010. (Cited on pages 20, 34, 54, and 55.)

[59] D. Ortega and P. Braun, "Information, utility and bounded rationality," *Artificial General Intelligence*, pp. 269–274, 2011. (Cited on page 6.)

[60] L. Hansen and T. Sargent, *Robustness*. Princeton, NJ: Princeton University Press, 2007. (Cited on pages 6, 19, 24, and 54.)

[61] E. Todorov and W. Li, "A generalized iterative LQG method for locally-optimal feedback control of constrained nonlinear stochastic systems," in *Proc. American Control Conference*, pp. 300–306, IEEE, 2005. (Cited on pages 7, 17, 48, 71, and 73.)

[62] J. Hill and W. T. Park, "Real-time control of a robot with a mobile camera," in *Proc. 9th International Symposium on Industrial Robots*, (Washington D.C, USA), pp. 233–246, March 1979. (Cited on pages 7 and 58.)

[63] F. Janabi-Sharifi and M. Ficocelli, "Formulation of radiometric feasibility measures for feature selection and planning in visual servoing," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 34, pp. 978–987, April 2004. (Cited on page 7.)

[64] F. Janabi-Sharifi and M. Marey, "A kalman-filter-based method for pose estimation in visual servoing," *Robotics, IEEE Transactions on*, vol. 26, pp. 939–947, October 2010. (Not cited.)

[65] F. Janabi-Sharifi, L. Deng, and W. Wilson, "Comparison of basic visual servoing methods," *Mechatronics, IEEE/ASME Transactions on*, vol. 16, pp. 967 –983, oct. 2011. (Not cited.)

[66] L. Deng, F. Janabi-Sharifi, and W. Wilson, "Hybrid motion control and planning strategies for visual servoing," *IEEE Transactions on Industrial Electronics*, vol. 52, pp. 1024 – 1040, aug. 2005. (Cited on page 7.)

[67] P. Whittle, *Optimal control: basics and beyond*. Hoboken, NJ: John Wiley & Sons, Inc., 1996. (Cited on page 13.)

[68] S. Marcus, E. Fernández-Gaucherand, D. Hernández-Hernandez, S. Coraluppi, and P. Fard, "Risk sensitive markov decision processes," *Progress in Systems and Control Theory*, vol. 22, pp. 263–280, 1997. (Cited on pages 14 and 43.)

[69] M. Zyskowski and R. Diersing, "Infinite-horizon, multiple-cumulant cost density-shaping for stochastic optimal control," in *Proc. American Control Conference*, pp. 1488–1493, IEEE, 2011. (Cited on page 14.)

[70] M. Zyskowski, M. Sain, and R. Diersing, "Weighted least-squares, cost density-shaping, stochastic optimal control: A step towards total probabilistic control design," in *Proc. 49th IEEE Conference on Decision and Control*, pp. 1417–1422, IEEE, 2010. (Cited on page 14.)

[71] B. Lindoff, J. Holst, and B. Wittenmark, "Analysis of approximations of dual control," *International Journal of Adaptive Control and Signal Processing*, vol. 13, no. 7, pp. 593–620, 1999. (Cited on page 16.)

[72] D. Bertsimas, D. B. Brown, and C. Caramanis, "Theory and applications of robust optimization," *SIAM Review*, vol. 53, no. 3, pp. 464–501, 2011. (Cited on pages 16 and 24.)

[73] M. G. Lagoudakis and R. Parr, "Least-squares policy iteration," *The Journal of Machine Learning Research*, vol. 4, pp. 1107–1149, 2003. (Cited on page 17.)

[74] E. Theodorou, Y. Tassa, and E. Todorov, "Stochastic differential dynamic programming," in *Proc. American Control Conference*, pp. 1125–1132, IEEE, 2010. (Cited on page 17.)

[75] G. N. Iyengar, "Robust dynamic programming," *Mathematics of Operations Research*, vol. 30, no. 2, pp. 257–280, 2005. (Cited on pages 18 and 24.)

[76] A. Nilim and L. El Ghaoui, "Robust control of markov decision processes with uncertain transition matrices," *Operations Research*, vol. 53, no. 5, pp. 780–798, 2005. (Cited on pages 18 and 24.)

[77] J. Yedidia, W. Freeman, and Y. Weiss, "Understanding belief propagation and its generalizations," *Exploring Artificial Intelligence in the New Millennium*, vol. 8, pp. 236–239, 2003. (Cited on pages 19, 40, 42, and 56.)

[78] A. Iusem, B. Svaiter, and M. Teboulle, "Entropy-like proximal methods in convex programming," *Mathematics of Operations Research*, vol. 19, no. 4, pp. 790–814, 1994. (Cited on pages 20, 27, and 28.)

[79] R. Rockafellar, "Augmented lagrangians and applications of the proximal point algorithm in convex programming," *Mathematics of Operations Research*, vol. 1, no. 2, pp. 97–116, 1976. (Cited on pages 20, 27, and 28.)

[80] E. A. Feinberg, "Optimality of deterministic policies for certain stochastic control problems with multiple criteria and constraints," in *Mathematical Control Theory and Finance*, pp. 137–148, Berlin, Heidelberg: Springer, 2008. (Cited on page 21.)

[81] M. M. Fard and J. Pineau, "MDPs with non-deterministic policies," *Advances in Neural Information Processing Systems*, vol. 21, p. 1065, 2009. (Cited on page 21.)

[82] S. Filippi, O. Cappé, and A. Garivier, "Optimism in reinforcement learning and kullback-leibler divergence," in *Proc. 48th Annual Allerton Conference on Communication, Control, and Computing (Allerton), 2010*, pp. 115–122, IEEE, 2010. (Cited on pages 23 and 26.)

[83] R. I. Brafman and M. Tennenholtz, "R-MAX-a general polynomial time algorithm for near-optimal reinforcement learning," *The Journal of Machine Learning Research*, vol. 3, pp. 213–231, 2003. (Cited on pages 23 and 26.)

[84] A. Ben-Tal and A. Nemirovski, "Robust solutions of linear programming problems contaminated with uncertain data," *Mathematical Programming*, vol. 88, no. 3, pp. 411–424, 2000. (Cited on page 24.)

[85] A. Ben-Tal, D. den Hertog, A. De Waegenaere, B. Melenberg, and G. Rennen, "Robust solutions of optimization problems affected by uncertain probabilities," *Management Science*, vol. 59, no. 2, pp. 341–357, 2013. (Cited on page 24.)

[86] H. David, C. Strauss, and D. Rajnarayan, "Advances in distributed optimization using probability collectives," *Advances in Complex Systems*, vol. 9, no. 04, pp. 383–436, 2006. (Cited on page 27.)

[87] D. Rajnarayan, D. Wolpert, and I. Kroo, "Optimization under uncertainty using probability collectives," in *Proc. 12th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, (Victoria, British Columbia, Canada.), 2006. (Cited on page 27.)

[88] A. Iusem, "Augmented lagrangian methods and proximal point methods for convex optimization," *Investigación Operativa*, vol. 8, pp. 11–49, 1999. (Cited on page 27.)

[89] M. Teboulle, "Entropic proximal mappings with applications to nonlinear programming," *Mathematics of Operations Research*, vol. 17, no. 3, pp. 670–690, 1992. (Cited on pages 27 and 28.)

[90] J. Bierkens and H. J. Kappen, "Probabilistic solution of relative entropy weighted control," *pre-print*, 2012. arXiv:1205.6946 [math.PR]. (Cited on page 28.)

[91] D. P. Bertsekas, "Multiplier methods: a survey," *Automatica*, vol. 12, no. 2, pp. 133–145, 1976. (Cited on page 29.)

[92] L. Hansen and T. Sargent, "Robust control and model uncertainty," *American Economic Review*, pp. 60–66, 2001. (Cited on pages 30 and 54.)

[93] L. Hansen, T. Sargent, G. Turmuhambetova, and N. Williams, "Robust control and model misspecification," *Journal of Economic Theory*, vol. 128, no. 1, pp. 45–90, 2006. (Cited on page 30.)

[94] J. Yedidia, W. Freeman, and Y. Weiss, "Constructing free-energy approximations and generalized belief propagation algorithms," *IEEE Transactions on Information Theory*, vol. 51, no. 7, pp. 2282–2312, 2005. (Cited on pages 33 and 56.)

[95] M. Wainwright and M. Jordan, "Graphical models, exponential families, and variational inference," *Foundations and Trends® in Machine Learning*, vol. 1, no. 1-2, pp. 1–305, 2008. (Cited on page 33.)

[96] J. Bezdek and R. Hathaway, "Some notes on alternating optimization," *Advances in Soft Computing–AFSS 2002*, pp. 187–195, 2002. (Cited on pages 37, 40, and 56.)

[97] H. Cox, "On the estimation of state variables and parameters for noisy dynamic systems," *IEEE Transactions on Automatic Control*, vol. 9, no. 1, pp. 5–12, 1964. (Cited on page 44.)

[98] E. Wan and R. Van der Merwe, "The unscented kalman filter for nonlinear estimation," in *Proc. Adaptive Systems for Signal Processing, Communications, and Control Symposium*, pp. 153–158, 2000. (Cited on page 44.)

[99] R. Van Der Merwe, A. Doucet, N. De Freitas, and E. Wan, "The unscented particle filter," *Advances in Neural Information Processing Systems*, pp. 584–590, 2001. (Cited on page 44.)

[100] T. P. Minka, "Expectation-propagation for approximate bayesian inference," in *Proc. 17th Conference on Uncertainty in Artificial Intelligence*, pp. 362–369, Morgan Kaufmann Publishers Inc., 2001. (Cited on page 44.)

[101] H. Kappen, "Path integrals and symmetry breaking for optimal control theory," *Journal of Statistical Mechanics: Theory and Experiment*, no. 11, pp. P110–121, 2005. (Cited on page 53.)

[102] K. Dvijotham and E. Todorov, "Linearly solvable markov games," in *Proc. American Control Conference*, pp. 1845–1850, IEEE, 2012. (Cited on page 54.)

[103] I. Petersen, M. James, and P. Dupuis, "Minimax optimal control of stochastic uncertain systems with relative entropy constraints," *Automatic Control, IEEE Transactions on*, vol. 45, no. 3, pp. 398–412, 2000. (Cited on pages 55 and 91.)

[104] A. Ypma and T. Heskes, "Iterated extended kalman smoothing with expectation-propagation," in *Proc. IEEE 13th Workshop on Neural Networks for Signal Processing*, pp. 219–228, 2003. (Cited on page 57.)

[105] P. Corke, "Visual control of robot manipulators - a review," in *Visual Serving: Real -Time Control of Robot Manipulators Based on Visual Sensory Feedback* (K. Hashimoto, ed.), vol. 7, pp. 1–31, Toh Tuck Link, Singapore: World Scientific Publishing Co, 1993. (Cited on pages 58, 59, and 65.)

[106] N. Papanikolopoulos, P. Khosla, and T. Kanada, "Vision and control techniques for robotic visual tracking," in *Proc. IEEE International Conference on Robotics and Automation*, pp. 857–864, IEEE, 1991. (Not cited.)

[107] F. Chaumette and S. Hutchinson, "Visual servo control. i. basic approaches," *IEEE Robotics & Automation Magazine*, vol. 13, no. 4, pp. 82–90, 2006. (Cited on pages 58, 59, and 64.)

[108] F. Chaumette and S. Hutchinson, "Visual servo control, part ii: Advanced approaches," *IEEE Robotics & Automation Magazine*, vol. 14, no. 1, pp. 109–118, 2007. (Not cited.)

[109] S. Hutchinson, G. Hager, and P. Corke, "A tutorial on visual servo control," *Robotics and Automation, IEEE Transactions on*, vol. 12, no. 5, pp. 651–670, 1996. (Cited on pages 58, 59, 64, and 65.)

[110] M. Sauvee, P. Poignet, E. Dombre, and E. Courtial, "Image based visual servoing through nonlinear model predictive control," in *Proc. 45th IEEE Conference on Decision and Control*, pp. 1776–1781, 2006. (Cited on pages 59, 66, and 72.)

[111] G. Allibert, E. Courtial, and F. Chaumette, "Predictive control for constrained image-based visual servoing," *IEEE Transactions on Robotics and Automation*, vol. 26, no. 5, pp. 933–939, 2010. (Cited on pages 59, 66, and 72.)

[112] G. Chesi and K. Hashimoto, *Visual Servoing Via Advanced Numerical Methods*. Lecture Notes in Control and Information Sciences, Springer, 2010. (Not cited.)

[113] G. Allibert, E. Courtial, and F. Chaumette, "Visual servoing via nonlinear predictive control," in *Visual Servoing via Advanced Numerical Methods* (G. Chesi and K. Hashimoto, eds.), vol. 401 of *Lecture Notes in Control and Information Sciences*, pp. 375–393, Berlin, Germany: Springer, 2010. (Cited on pages 59 and 66.)

[114] B. Siciliano and L. Sciavicco, *Robotics: modelling, planning and control*. Berlin, Germany: Springer, 2009. (Cited on pages 60 and 75.)

[115] D. Lowe, "Object recognition from local scale-invariant features," in *Proc. 7th IEEE International Conference on Computer Vision*, vol. 2, pp. 1150–1157, Sept 1999. (Cited on page 61.)

[116] F. Hoffmann, T. Nierobisch, T. Seyffarth, and G. Rudolph, "Visual servoing with moments of SIFT features," in *Proc. IEEE International Conference on Man and Cybernetics.*, vol. 5, pp. 4262–4267, IEEE, 2006. (Cited on page 61.)

[117] J. Heikkila and O. Silven, "A four-step camera calibration procedure with implicit image correction," in *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, (Puerto Rico), pp. 1106–1112, IEEE, 1997. (Cited on pages 61 and 75.)

[118] F. Chaumette, P. Rives, and B. Espiau, "The task function approach applied to vision-based control," in *Proc. 5th International Conference on Advanced Robotics*, (Washington, D. C), pp. 1392–1397, IEEE, 1991. (Cited on pages 63 and 64.)

[119] P. Rives, F. Chaumette, and B. Espiau, "Visual servoing based on a task function approach," in *Experimental Robotics I* (V. Hayward and O. Khatib, eds.), vol. 139 of *Lecture Notes in Control and Information Sciences*, pp. 412–428, Springer, 1990. (Cited on page 63.)

[120] F. Chaumette, "Potential problems of stability and convergence in image-based and position-based visual servoing," in *The Confluence of Vision and Control* (D. Kriegman, G. Hager, and A. Morse, eds.), vol. 237 of *Lecture Notes in Control and Information Sciences*, pp. 66–78, Springer London, 1998. (Cited on pages 66 and 80.)

[121] K. Hashimoto, T. Ebine, and H. Kimura, "Visual servoing with hand-eye manipulator-optimal control approach," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 5, pp. 766–774, 1996. (Cited on page 66.)

[122] M. Sznaier and O. Camps, "Control issues in active vision: open problems and some answers," in *Proc. 37th IEEE Conference on Decision and Control*, vol. 3, pp. 3238–3244 vol.3, 1998. (Cited on page 66.)

[123] Y. Bar-Shalom and E. Tse, "Dual effect, certainty equivalence, and separation in stochastic control," *IEEE Transactions on Automatic Control*, vol. 19, no. 5, pp. 494–500, 1974. (Cited on pages 68 and 69.)

[124] Y. Bar-Shalom and E. Tse, "Caution, probing, and the value of information in the control of uncertain systems," in *Annals of Economic and Social Measurement*, vol. 5, pp. 323–337, NBER, 1976. (Not cited.)

[125] A. Feldbaum, "Dual control theory i," *Automation and Remote Control*, vol. 21, no. 9, pp. 874–1039, 1960. (Cited on page 68.)

[126] D. P. Joseph and T. J. Tou, "On linear control theory," *American Institute of Electrical Engineers, Part II: Applications and Industry, Transactions of the*, vol. 80, no. 4, pp. 193–196, 1961. (Cited on page 68.)

[127] E. Tse, "Adaptive dual control methods," in *Annals of Economic and Social Measurement*, vol. 1, pp. 65–84, 1974. (Cited on page 68.)

[128] H. Unbehauen, "Adaptive dual control systems: a survey," in *Proc. IEEE Adaptive Systems for Signal Processing, Communications, and Control Symposium*, pp. 171–180, IEEE, 2000. (Cited on page 69.)

[129] B. Wittenmark, "Adaptive dual control methods: An overview," in *Proc. 5th IFAC symposium on Adaptive Systems in Control and Signal Processing*, Citeseer, 1995. (Cited on page 68.)

[130] A. De Luca, G. Oriolo, and P. Robuffo Giordano, "Feature depth observation for image-based visual servoing: Theory and experiments," *The International Journal of Robotics Research*, vol. 27, no. 10, pp. 1093–1116, 2008. (Cited on page 69.)

[131] A. Davison, I. Reid, N. Molton, and O. Stasse, "Monoslam: Real-time single camera slam," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 29, no. 6, pp. 1052–1067, 2007. (Not cited.)

[132] J. Civera, A. J. Davison, and J. Montiel, "Inverse depth parametrization for monocular slam," *Robotics, IEEE Transactions on*, vol. 24, no. 5, pp. 932–945, 2008. (Cited on page 69.)

[133] A. Bemporad and M. Morari, "Robust model predictive control: A survey," in *Robustness in Identification and Control* (A. Garulli and A. Tesi, eds.), vol. 245 of *Lecture Notes in Control and Information Sciences*, pp. 207–226, Springer London, 1999. (Cited on page 72.)

[134] G. Allibert, E. Courtial, and Y. Toure, "Visual predictive control for manipulators with catadioptric camera," in *Proc. IEEE International Conference on Robotics and Automation*, pp. 510–515, 2008. (Cited on page 72.)

[135] Thermo-CRS, *A255 Robot System User Guide UMI-A255-400*. Thermo Scientific, Mississauga, ON, Canada, 002a ed., July 2002. (Cited on page 75.)

[136] Quanser, *QuaRC CRS CataLyst-5T Open-Architecture Manual*. Quanser Inc., Markham, Ontario, Canada, 2008. (Cited on page 75.)

[137] Point Grey, Richmond, BC, Canada, *Camera Model: FFMV-03M2M-CS*. http://ww2.ptgrey.com/IEEE-1394/fireflymv. (Cited on page 75.)

[138] N. D. Inc., *Optotrak Certus Motion Capture System*. Northern Digital Inc., Waterloo, ON, Canada. (Cited on page 75.)

[139] P. I. Corke, *Robotics, Vision & Control: Fundamental Algorithms in Matlab*. Berlin, Germany: Springer, 2011. (Cited on page 80.)

[140] C. Charalambous and F. Rezaei, "Stochastic uncertain systems subject to relative entropy constraints: Induced norms and monotonicity properties of minimax games," *Automatic Control, IEEE Transactions on*, vol. 52, no. 4, pp. 647–663, 2007. (Cited on page 91.)

[141] J. Gonzalez, Y. Low, and C. Guestrin, *Scaling Up Machine Learning*, ch. Parallel Belief Propagation in Factor Graphs, pp. 190–217. Cambridge, UK: Cambridge University Press, 2012. (Cited on page 91.)

[142] N. Ma, Y. Xia, and V. Prasanna, "Data parallelism for belief propagation in factor graphs," in *Proc. 23rd International Symposium on Computer Architecture and High Performance Computing*, pp. 56–63, 2011. (Cited on page 91.)