MODELING AND CONTROL FOR ROBOTIC ASSISTANTS: SINGLE AND MULTI-ROBOT MANIPULATION

Monroe D. Kennedy III

A DISSERTATION

in

Mechanical Engineering and Applied Mechanics

Presented to the Faculties of the University of Pennsylvania

in

Partial Fulfillment of the Requirements for the

Degree of Doctor of Philosophy

2019

Dedicated to my loving family

Acknowledgments

I would first like to thank my advisor Vijay Kumar for the opportunity to work in his world-class laboratory with amazing collaborators and exciting, multi-disciplinary research. I truly appreciate his mentorship technique, throughout my studies he always had high expectations which he coupled with support. This led me to challenge myself and rise to the research objectives he initially set for me, and later, to those I would set for myself. By observing and working with him, I learned how to have academic confidence combined with techniques to quickly assimilate knowledge. Not only did he guide me through fundamentals, but also showed me how to confidently address new research directions with sound research skills. Being a part of multiple projects, I was able to observe first hand how he seamlessly transitioned between very different projects in aerial vehicles, micro-robots and manipulation while remaining extremely insightful and guiding directions for research. His multi-disciplinary ability and lack of fear when it came to addressing new problems is something I will strive to reproduce for my entire career. His care for myself and my labmates made the lab a family, and I will do my very best to continue his legacy of academic and professional excellence.

I would like to thank my co-advisor Kostas Daniilidis for his wisdom and guidance as he provided me with real-world research problems to tackle. He also provided me with access to his robots and gave me a deep appreciation for modern techniques in robotics which I now combine with classical approaches. I would like to thank Dinesh Thakur for showing me how to connect theory with practical implementation, and for working alongside me as we completed papers and competitions. I would like to thank my committee members, Dr. M. Ani Hsieh, Dr. Mark Yim and Dr. Subhrajit Bhattacharya of Lehigh University, for all of their guidance and for their collaboration on papers. I would like to thank my lab-mates for filling these few years with laughter, and for holding each other up when things were intense. I would like to thank the MEAM and GRASP Staff for their unfailing support.

I would like to thank my parents for giving me a love for engineering at a very young age. My dad, for providing me with a template of excellence both as an academic and as an individual. My mom, for sacrificing her career to directly invest her engineering and physics degrees in my siblings and my education. I would like to thank my wife Sonia, for always caring for me, praying for me and building me up on this journey, I cross the finish line standing and strong because of her and answered prayers.

I would like to thank the National Consortium for Graduate Degrees for Minorities in Engineering and Science (GEM) and the National Science Foundation (NSF) for their support through graduate research fellowships. I am grateful to NSF for their support through grants NSF-IIS1138847, NSF-IIS1328805, NSF-IIS1426840, NSF-1138847, NSF-1328805 and also to GlaxoSmithKline through NSF-IIP-1439681. I would also like to thank Army Research Laboratory for their support through W911NF-10-2-0016 in the Robotics Collaborative Technology Alliance.

ABSTRACT

MODELING AND CONTROL FOR ROBOTIC ASSISTANTS:

SINGLE AND MULTI-ROBOT MANIPULATION

Monroe D. Kennedy III

Kostas Daniilidis

Vijay Kumar

As advances are made in robotic hardware, the complexity of tasks they are capable of performing also increases. One goal of modern robotics is to introduce robotic platforms that require very little augmentation of their environments to be effective and robust. Therefore the challenge for a roboticist is to develop algorithms and control strategies that leverage knowledge of the task while retaining the ability to be adaptive, adjusting to perturbations in the environment and task assumptions. This work considers approaches to these challenges in the context of a wet-lab robotic assistant. The tasks considered are cooperative transport with limited communication between team members, and robot-assisted rapid experiment preparation requiring pouring reagents from open containers useful for research and development scientists. For cooperative transport, robots must be able to plan collision-free trajectories and agree on a final destination to minimize internal forces on the carried load. Robot teammates are considered, where robots must reach consensus to minimize internal forces. The case of a human leader, and robot follower is then considered, where robots must use non-verbal information to estimate the human leader's intended pose for the carried load. For experiment preparation, the robot must pour precisely from open containers with known fluid in a single attempt. Two scenarios examined are when the geometries of the pouring and receiving containers and behaviors are known, and when the pourer must be approximated. An analytical solution is presented for a given geometry in the first instance. In the second instance, a combination of online system identification and leveraging of model priors is used to achieve the precision-pour in a single attempt with considerations for long-term robot deployment. The main contributions of this work are considerations and implementations for making robots capable of performing complex tasks with an emphasis on combining model-based and data-driven approaches for best performance.

Contents

List of Tables

List of Figures

Nomenclature

SE(3) The special euclidean group in three dimensions.

W, R, E, H, O SE(3) frames: world, robot, end-effector, human, carried load (respectively).

A, B Mobile manipulator arm and base SE(3) frames respectively.

t, t_d, t_0, t_f Time: t is current time, t_d is time delay, t_0 is an initial time and t_f is the final time.

$R_{\mathrm{W}A}$ Rotation matrix: subscript indicates rotation from frame A to W.

$\boldsymbol{\phi}_{\mathrm{W}A}$ Vector of exponential coordinates: rigid body orientation vector in SE(3). Subscript indicates rotation from frame A to W. $\boldsymbol{\phi}_{\mathrm{W}A}$ is the axis angle representation of rotation from frame A to W which is the unit rotation axis multiplied by the rotation angle and satisfies the exponential map: $R_{\mathrm{W}A} = exp\left(\boldsymbol{\phi}_{\mathrm{W}A}^{[\times]}\right)$ (which is the exponential of the skew symmetric matrix constructed from $\boldsymbol{\phi}_{\mathrm{W}A}$).

$\boldsymbol{r}_i$ Position vector in inertial frame W.

$\boldsymbol{p}_{\mathrm{W}A}(t)$ Pose vector: the pose of frame A in inertial frame W, $\boldsymbol{p}_{\mathrm{W}A}(t) = \begin{bmatrix} x & y & z & \phi_x & \phi_y & \phi_z \end{bmatrix}^T$ with x, y, z being the translation component and $\boldsymbol{\phi}_{\mathrm{W}A}$ the vector of exponential coordinates.

$T(\boldsymbol{p}_{\mathrm{W}A}), T_{\mathrm{W},A,} FK(\,\cdot\,)$ Transformation matrix: SE(3) transformation matrix describing translation and rotation from frame A to frame W. $FK(\,\cdot\,)$ is transformation from inertial frame to end-effector frame.

$\boldsymbol{d}$ Translation component x, y, z of rigid body transformation.

x, $\dot{x}$, $\hat{x}$, $\bar{x}$, $\boldsymbol{x}$, X State representations: state, time differentiated state, estimated state, state mean, state vector, state matrix respectively.

z Output variable, derived from the state.

$\boldsymbol{v}$, $\boldsymbol{\omega}$ Velocities: Linear and angular velocities in inertial frame.

$q_{a,i}$, $q_{b,i}$, $\boldsymbol{q}$ Generalized coordinates: generalized coordinates of manipulator, base and vector representation.

$J(q)$ Jacobian: The derivative of the forward kinematics map with respect to generalized coordinates.

m, ρ, V Material properties: mass, density and volume.

g Gravity.

l_i Manipulator link lengths: (constant quantities for determining forward kinematics).

$\mathcal{I}$, $\tilde{\mathcal{I}}$ Rigid body inertia: 3×3 inertia matrix in body fixed frame and transformed into another frame.

$D(\boldsymbol{q})$, $C(\dot{\boldsymbol{q}},\boldsymbol{q})$, $G(\boldsymbol{q})$ System Dynamical Terms: Generalized inertia matrix, Coriolis, centrifugal vector and gravity vector.

$\boldsymbol{\pi}$ Vector for linearized manipulator dynamics.

$\mathcal{W}$ Work.

$\mathcal{K}, \mathcal{P}$ Energy: kinetic and potential energy respectively.

$\mathcal{L}$ Lagrangian.

$\mathcal{H}$ Hamiltonian.

$\boldsymbol{\tau}$ Torque: applied at revolute joints.

$\boldsymbol{f}$ 3x1 force vector in inertial frame.

$\boldsymbol{w}$ Wrench: 6x1 force and torque vector in inertial frame

$\boldsymbol{e}_x$ Error vector, subscript indicates state.

C_h, C_{nh} Sets for holonomic and non-holonomic constraints.

A_{manip} Constraint matrix for manipulator dynamics against external forces.

G_i Contact wrench: contact wrench used to define the resultant force and moment in body fixed frame for a force applied at $\boldsymbol{r}_i$.

$f(\,\cdot\,)$, F Dynamical systems model: describes the evolution of states as a function of the current state, inputs and systems parameters. F is a linear model.

$\boldsymbol{\beta}(t)$ Model parameter vector: vector of model parameters that may vary with time.

S Objective function for optimization.

P_H Objective function hessian.

∇_x Gradient operator.

γ Step-size gain.

λ Lagrange multipliers.

Q Positive semi-definite matrix.

$pr(\,\cdot\,)$ Probability density function.

p_{stat} Discrete probability of an event.

η Noise: usually associated with sensor measurement or general uncertainty.

$\mathcal{N}$ Gaussian Distribution

Σ Generic covariance matrix.

χ Chi-points: points in unscented Kalman filter used to characterize the uncertainty for nonlinear functions

$Z_{kf}, \chi, K_{kf}, Q_{kf}, R_{kf}, L_{kf}, M_{kf}$ Kalman Filter: measurement matrix, Chi/sigma points, Kalman gain, measurement uncertainty, process uncertainty, measurement covariance update, state covariance update.

$\sigma(\,\cdot\,)$, $cov(\,\cdot\,)$, Γ_{kf}, W_σ Kalman filter: sigma point operator, covariance operator, sigma point matrix operator, sigma point weighting matrix.

$\kappa(\,\cdot\,,\,\cdot\,)$ Kernel function.

$V[\,\cdot\,]$ Variance: used to characterize uncertainty of a test point for a Gaussian process.

σ_κ, l_κ,K, L_{GP} Gaussian process parameters: Gaussian radial basis function variance and length-scale, covariance matrix, information matrix.

e_{res} Residual error between model approximation and generating data.

u Control input.

$\boldsymbol{s}$ Control objective: error attenuation after feedback linearization if necessary.

$\mathcal{L}_{lie}$ Lie derivative operator: lie derivative of a function along a vector field measures changes in the function along differential equation solutions of the vector field.

$\mathcal{V}$, $\mathcal{L}_{lie,f}\mathcal{V}(\boldsymbol{x})$ Lyapunov function and Lie derivative, note that if $\boldsymbol{x}(t)$ is a solution to $\dot{\boldsymbol{x}}(t) = f(\boldsymbol{x}(t), t)$ then $\dot{\mathcal{V}}(\boldsymbol{x}(t)) = \mathcal{L}_{lie,f}\mathcal{V}(\boldsymbol{x})$.

M, Ω_c, S_s Controller Stability: invariant set, region of stability, stable set.

K_p, K_d Control gains: proportional and derivative gains.

$\mathbb{L}$, $\mathbb{A}$, $\mathbb{D}$, $\mathbb{G}$	Laplacian matrix, adjacency matrix, diagonal neighbor matrix and graph respectively.
T_{chrd}	Path Planning: time allocated to complete path chord segment.
$\boldsymbol{h}$, $\boldsymbol{s}$	Human Intent Quantities: Human leader head orientation and step vectors.
α, β	Pouring and receiving containers respectively. These will be used in Part III.
V_α, V_β	Container volume: for either container α or β.
A_β, A_α	Area: associated with cross-sections in either containers α or β.
h_α, h_β	Fluid height: fluids heights associated with either containers α or β.
$L_{L,\alpha}$	Container lip width function: describes lip width for the pouring container.
q_{fr}	Volumetric flowrate.
V_β	Pouring: receiving container fluid volume.
$V_{L,\alpha}$,$V_{\alpha,t}$	Pouring: volume above the pouring container lip edge.
$V_{s,\alpha}$, $V_{\alpha,\theta}$	Pouring: volume below the pouring container lip edge.
$A_{s,\alpha}$	Pouring: cross sectional area separating volume above and below pouring edge in pouring container.
$h_{L,\alpha}$	Pouring: height of fluid above the pouring edge lip of the pouring container.
A_L	Pouring: cross-sectional area of fluid exiting pouring container at the pouring edge.
$v(h_{L,\alpha})$	Pouring: velocity of fluid exiting pouring container.
P_{bern}	Pouring: Pressure on fluid from atmosphere used in Bernoulli equation.
W_α	Pouring: Specified pouring container width.
H_α	Pouring: Specified pouring container height.
l_α	Pouring: Specified pouring container length.
φ, $r_{L,\alpha}$	Pouring: pouring container lip opening angle for V-shaped lip with angle φ and circular lip with radius $r_{L,\alpha}$.
$\Gamma(\,\cdot\,)$	Pouring: Symmetric container geometry function, describes symmetric radius as a function of height.
H_Γ	Pouring: Symmetric container height.
$V_{\beta,\theta}$, $V_{\alpha,\theta}$	Pouring: Volume profile, this is the volume versus angle for the receiver and pouring containers.
e_Γ	Pouring: 'distance' between two symmetric container geometries.
$\bar{r}_{res}$, $r_{res,max}$	Pouring: residual after fitting volume profile, residual average and maximum allowed.

Part I

Preliminaries

Chapter 1

Introduction

1.1 Motivation

Over the past few decades robotics has evolved from automation in the 1980-90's to major breakthroughs in autonomy in the 2000's and now to human-augmentation. With each advance, robots have become more capable, with augmentation being the most exciting endeavor to date. The goal is to now go beyond teleoperation and performing repetitive tasks, and become effective, capable collaborators with human teammates. To meet this objective robots must not only be capable of completing complex tasks, but it is essential that the robots be able to consider humans working alongside them. Using human-human collaboration as a standard, humans are able to effectively complete complex tasks together, and sometimes require minimal explicit (e.g. verbal) communication by anticipating the actions of others in the team. Incorporating these abilities into robots is a large area of research, and this work takes a step towards making it become a reality.

1.2 Challenges and Opportunities

To be effective collaborators, robots must be able to perform complex tasks reliably in unstructured, dynamic environments designed for humans. An outstanding challenge is how to effectively model the complex tasks in such a way that natural perturbations that occur for a deployable robotic system can be accommodated within expected operation conditions. Another challenge is how can the robot become a more effective collaborator, with the ability to communicate in more natural ways with human counterparts (where natural indicates similarity to human-human interaction). An additional challenge is how to make the robot capable of performing complex tasks and becoming an effective collaborator with limited training examples and data. To address these challenges, this work proposes to leverage model-based approaches (such as knowledge of physical models and principles) along with the modern data-driven techniques in system identification, to allow the robot to adapt to perturbations in task assumptions and in one presented instance, even estimate the humans intent within the task.

1.3 Problem Statement

Given a collaborative (or assistive) manipulation task with a given metric of success, can the robot's model of the task be learned or adapted, and a controller designed which maximizes the metric of success?

For a given task which can be represented in terms of relevant states $\boldsymbol{x}$, the problem is to determine the set of robot actions u that will facilitate the desired state. Formally, given a state $\boldsymbol{x}(t)$ which combines the robot and the task, the robot's input $u(t)$, modeling parameters $\boldsymbol{\beta}(t)$, and output $\boldsymbol{z}(t)$, the general system dynamics are described by

$$\begin{aligned} \dot{\boldsymbol{x}}(t) &= f(\boldsymbol{x}(t), u(t), \boldsymbol{\beta}(t), t) \\ \boldsymbol{z}(t) &= h(\boldsymbol{x}(t), \boldsymbol{\beta}(t), t) \end{aligned} \tag{1.1}$$

where $f(\,\cdot\,)$ is the dynamical system process model and $h(\,\cdot\,)$ is the output function. If a desired output based on the task objective is described by the function

$$\boldsymbol{z}_{des}(t) = h_{des}(\boldsymbol{x}(t), \boldsymbol{\beta}(t), t), \tag{1.2}$$

then output error can be defined as

$$\boldsymbol{e}(t) = \boldsymbol{z}_{des}(t) - \boldsymbol{z}(t) \tag{1.3}$$

The goal is to find the control input which meets the task objective by minimizing this error while better approximating the system dynamics. In certain applications this requires adjusting the model of the system dynamics to better reflect the true world model. This is represented formally by minimizing the following functional subject to the system dynamics

$$\begin{aligned} \min_{u(t)} S &= m(\boldsymbol{x}(t_f), u(t_f), t_f) + \int_{t_0}^{t_f} g(\boldsymbol{e}(t), \boldsymbol{\beta}(t), \boldsymbol{x}(t), u(t), t)dt \\ \textit{subj to: } \dot{\boldsymbol{x}}(t) &= f(\boldsymbol{x}(t), u(t), \boldsymbol{\beta}(t), t) \\ \boldsymbol{z}(t) &= h(\boldsymbol{x}(t), \boldsymbol{\beta}(t), t) \end{aligned} \tag{1.4}$$

where $m(\cdot)$ and $g(\cdot)$ are the terminal and running costs respectively. For the described model and parameters this may reduce to an optimal control problem, but for unknown model parameters this formally considered an adaptive control problem. In some instances the complete structure of $f(\cdot)$ and $h(\cdot)$ are unknown (not just parameters $\boldsymbol{\beta}$). In this case to meet the overall objective by reaching a final desired output, these functions must also be identified.

1.4 Thesis Contributions and Outline

Figure 1.1: An illustration of a mobile manipulator. This manipulator has seven degrees of freedom and the mobile base is non-holonomic. For every joint the position, velocity and efforts are observable.

When a robot performs a manipulation task, the control problem in (1.4) can be subdivided into the characterization of a functional describing operational success, a model for the robot, a model for the task and a control policy. The control policy is designed to combine the knowledge of both the robot and the task to approach the optimum described by the functional. When prior knowledge about the task dynamics are known, it can be combined with system identification techniques for improved modeling. For many tasks, valid performance requires prediction of a future state of the system. When this is required, the estimated model can be used in predictive control with the performance improving with increasing model accuracy.

This thesis considers these concepts of task modeling for stable control in real-world collaborative tasks. This work first presents the cooperative transport problem where robots must transport a load as a member of a team through an obstacle filled environment. The task is to transport the load while avoiding obstacle collisions, as well as minimizing internal stresses in the carried load which consume energy and do not contribute to motion. It is noted that the optimal solution is centralized, where a single governor dictates the actions of all members. However when robots are decentralized, or when humans are present as teammates, centralized solutions are not always viable. Hence, it becomes a necessity to collaborate in order to approach the centralized performance. The contributions of this work toward this goal considers the entire pipeline of effective transport. First, if a goal pose is agreed upon by all teammates, with the assumptions that all collaborators want to avoid collision and conserve energy by taking the shortest feasible path this work presents a computational efficient way to decompose traversable free-space which contains the optimal path. Secondly, this work considers the case when robots are teammates with other robots while not knowing other robots grasp locations, but the the goal pose and optimal path is known by all robots. For this case, this work presents a computationally efficient way that robots can minimize internal stresses in the carried load if they reach consensus while simultaneously transporting. Third, this work considers the case when robots are teammates with humans, and humans are the leaders with an internal goal pose that is not initially known to the robot. In this case it is assumed that the human does not want to collide with obstacles and wants to take the shortest, feasible path. This work presents a way that robots can learn from how humans effectively transport with humans based on human leader observation, specifically asking the question: based what direction is the leader's head pointing, how are their feet moving, what is the haptic force and torque felt in the carried load applied by the leader, what is the velocity of the carried load and finally what are the surrounding obstacles how do human followers predict the motion of human leaders? A method of transferring this highly complex model from the human-human observation to the robot is successfully demonstrated. With this ability to reasonably predict the humans expected pose on a short time horizon, then within that short look-ahead time a collision free trajectory can be applied and with the known grasp points of the human and robot the wrench that minimizes internal stresses can also be applied along the calculated optimal trajectory for efficient transport. This work presents this method of prediction and leaves for future work the complete combination of the pipeline from demonstrated successful components.

This work then considers the task of a robot assistant preparing an experiment for collaboration with wet-laboratory research scientists. The task is to efficiently pour fluid between an open pouring and a receiving container. An efficient pour is characterized as both quick and precise. It is assumed that there is no spillage and that the properties of the poured fluid are known. The contributions of this work for this task are two fold. In the case where the geometry of all containers are known, and the receiver can be observed during the pour, an analytical solution is provided which considers the expected flow-rate of the fluid given both containers geometry. With this analytical solution, the robot is able to pour very quickly and precisely leveraging a hybrid controller that both ensures steady flow and considers the system dynamics in order achieve the desired receiver volume. In the case where the geometry of the pouring container is not known analytically, it is assumed that the geometry can be approximated by an initial external scan of the container. While for simplicity this work considers symmetric pouring containers, this is not a limitation as long as a distance metric can be defined between container geometries. During the pour only the receiving container is observed, and the system dynamics must be adjusted based on an initial estimate obtained from a single scan of the pouring container geometry. The main contribution in this case is that both classical system identification and learning are used simultaneously to predict the pouring container behavior. This prediction is a necessity because the time delay between the rotational velocity control input of the pouring container and the observed volume in the receiver is substantial for certain pouring container geometries. This predicted model for the pouring container is then used in a hybrid controller to achieve the control objective by anticipating the nonlinear dynamics. This is implemented on the Rethink Robotics Sawyer and KUKA intelligent industrial work assistant (iiwa) manipulators.

The methods developed in this work are easily generalized for new tasks. It is assumed that the task process model is not fully known, but some inherent structure can be exploited. For a process model manifold that is continuous with low curvature, system identification can leverage the inherent structure and be coupled with model priors from data utilized in a framework such as Bayesian modeling approach (in this work a Gaussian process). When the process model manifold is continuous but has high curvature, statistical modeling approaches like the Gaussian process become expensive in areas of high curvature as the generating data must be maintained. Here it is advantageous to utilize a neural network with adequate architecture. This is because the computational graph is capable of representing the highly curvature manifold of the process manifold with the static computational graph structure, allowing for an efficient model representation. Often with high curvature process model manifolds it is harder to exploit knowledge of the inherent structure. However this knowledge can usually be used as a guide when designing the neural net architecture.

The rest of the thesis is organized as follows, Chapter 2 presents related work in collaborative robotics and task adaptation, Chapter 3 presents mathematical preliminaries. In the next section, the problem of cooperative transport is investigated. Robots must collaborate in order to transport a load in a cluttered environment without collision while conserving energy by minimizing internal stresses on the load during transport. Given a target pose for the carried load, Chapter 4 presents a method of computationally efficiently modeling the free space that can be traversed. This free space decomposition is then used to obtain an optimal trajectory that remains in the interior of the free space. In Chapter 5, it is assumed that a target trajectory has been determined, and the goal is to efficiently transport the load while minimizing internal stresses and therefore conserve energy. When all robotic teammates know the grasp location of all other teammates the control input can be calculated to transport along the trajectory while minimizing internal stresses. Chapter 5 investigates the scenario where there are teams of robots, but communication is limited or expensive and the grasp location of other robotic teammates are unknown. An efficient consensus algorithm is proposed where robots share critical, minimal information in order to reduce internal stresses. In Chapter 6, a human leader robot follower scenario is considered. The stipulation is that explicit communication (verbal, digital, gestures) is not permissible between the human and robot. It is also assumed that no prior contextual knowledge about the carried object is known for the robot follower to make an educated guess on the expected final intended location. Therefore the robot must observe both the human leader and the surrounding environment to make an educated guess on a local horizon about the humans intended target pose which would lie on the global intended trajectory. Chapter 6 presents a method of leveraging select measurable quantities to estimate the human leaders intended pose on a local time horizon. *Together, these sections describe and provide contributions to the entire cooperative transport framework for human robot collaborative transport*. When carrying with a human collaborator, Chapter 6 allows for the prediction of the leaders intended pose for a local time horizon, Chapter 4 allows for computationally efficient free space decomposition for planning a collision free trajectory to the local goal, and if the grasp location of the human is known then Chapter 5 presents a controller to actuate along the trajectory in a manner that minimizes interaction forces.

In the next section, autonomous precision pouring is investigated. The goal is for a robotic assistant to quickly and precisely pour a specified amount of a known fluid between open containers. This is directly useful for wet-lab scientists who perform many rapid experiments. In Chapter 7 it is assumed that analytically models for both the pouring and receiving container geometries are known. A control strategy is proposed that allows for quick precise pours leveraging the known analytical model. In Chapter 8 the geometry of the pouring container does not have an analytical representation and the geometry must be estimated in order to approximate the underlying pouring dynamics. A modeling strategy is proposed which leverages both system identification as well as machine learning to estimate the model while performing control in order to pour precisely from the known container in a single attempt. Finally, conclusions and ideas for future work are presented in Chapter 9.

Chapter 2

Collaborative Robotics

2.1 Collaborative Tasks

The goal of robotics is to make human life safer, more enjoyable and more productive. As robotics matures use cases arise where robots must work with and around humans in close proximity while manipulating objects and navigating in unstructured, dynamic environments. Specifically, human-robot collaboration is an important emerging field with high societal and economic impact. Robots can perform collaborative tasks in a variety of environments, including but not limited to indoor environments such as a domestic or office buildings and perform service tasks for assisted living or general assistance. Assistant robots are also valuable in outdoor environments in both rural and urban settings with examples being street cleaning, farming or construction. This work will focus on robot assistants in indoor environments, specifically pertaining to home or work environments. An example of collaborative robots in such domestic settings is robots living alongside people as effective ‘roommates’, setting the stage for robots preparing meals and performing or assisting in chores [7, 8, 53, 84]. Collaborative robots can also be considered to directly augment the disabled, with a common use case being a semi-autonomous wheelchair with manipulation capabilities [42, 52]. In these instances, robots must be able to perform dexterous tasks in human-based unstructured environments. A challenge is the modeling the expected dynamics of manipulated objects while adapting to other environmental considerations that may perturb the expected behavior while robustly performing the given task. Manipulated objects generally have affordances (an intended use and style of manipulation) that may be a function of the object geometry or augmented property (e.g. an open container filled with liquid may be handled differently than an empty container). Appropriate methods to approximate and model these affordances for manipulation while achieving an expected performance is a topic of ongoing research [51].

2.2 Methods of Improving Task Execution in Unstructured Environments

One approach to improve manipulation abilities for complex tasks in low or unstructured environments is learning from demonstration. In the context of manipulation, various levels of abstraction may be employed. For instance, the relation between task success and joint motion or even low level joint velocities and efforts correlating to a portion of a task. In imitation learning, the robot attempts through trial and error to reproduce operational success by extracting key features from observed motions and efforts [36, 37, 53]. The robot may also record the joint states during successful operation and try to reproduce these for repeated success [57, 58].

A more contextual approach is to better understand the task and environment in order to better reason about the conditions leading to successful task execution. One obvious approach is to use prior knowledge about expected interaction during manipulation with a level of compliance so that desired performance can be achieved without a highly accurate model [1]. Another approach is to improve the model by explicitly involving the human in the model update through direct questions [12]. The robot can interact with the human in the context of training where the robot tests the humans response, essentially playing games with a purpose [36]. If it is possible for the robot to 'fail' at a task multiple times with the ability to reset the task state and not suffer significant repercussion, then the robot may perform the task and use failure cases to improve performance under certain assumptions [69].

One component the robot may be required to model is the collaborating human's intent. One effective way to convey intent is through explicit verbalization [84]. A drawback to this approach is that for complex tasks, this form of communication can quickly become laborious. When intent is not explicitly stated it is generally impossible for the robot to know with absolute certainty what the human intends [7]. Therefore, in such cases it is often useful to consider the human input in the context of the task and environment along with some prior knowledge. An example of this in the transport scenario is the assumption that the human would want to avoid collision. In this example, the task becomes determining the free region of space and estimating the desired path which would minimize expended energy [29, 52]. Methods of implicit communication may include (but are not limited to) tracking of the pose of the human leader's head, observing the wrench applied by the leader and observing the motion of the human leader's feet [7, 20, 67].

The works considered in this section all performed the tasks of autonomous pouring or cooperative transport in a cluttered environment in whole or in part. The next section will further consider the task of cooperative transport.

2.3 Cooperative Transport

The general cooperative transporting problem is to have a robotic agent work effectively with single or multiple counterparts to transport an object through a cluttered environment without collision while wasting limited energy on motions or forces that do not contribute to the specified desired trajectory. One way to characterize these wasted forces is through interaction forces, which are forces that apply tension or compression to the carried object while not contributing to overall wrench or motion [35, 78]. An example of wasted motion is that which does not contribute to achieving the objective trajectory or obstacle avoidance, adding no value to transport (e.g. spinning in place). When there is a designated leader, the robot must be able to ascertain the desired trajectory from the leader in order to carry effectively. This objective can be performed through explicit communication where for a human leader indicates desired motions verbally to direct the robot. And in the case of repeated carrying paths, the expected efforts can be recorded and mimicked [57, 84]. When explicit communication is limited, the robot teams can use their limited communication to augment existing shared knowledge to transport effectively [30, 31, 68, 71]. When explicit or digital communication is not possible or feasible, then implicit communication must be employed. A common approach is to allow the robot to be compliant, where an applied wrench over a given threshold causes the motion of the robot while it supports the carried object [1, 24–26, 67, 76]. In addition to compliance, it is important for the robot to move in a manner that considers the grasping and kinematic constraints [17, 63, 74, 82]. It is also important during fast motions to consider the coupled dynamics of the manipulator and the mobile base of the mobile robot [14, 83]. Another consideration is the modeling of obstacles for collision avoidance. Obstacles can be represented using polytopes with mixed integer quadratic programming to ensure estimated feasible trajectories lie outside obstacles [43]. This is usually very computationally expensive, which leads to the method of modeling free traversable space for collision avoidance [16, 29, 77], and is usually far more computationally tractable. Another approach is to use a more continuous representation of obstacles with potential fields, where viable paths do not pass through or too close to obstacles as the heuristic would drastically penalize an invalid path [52]. Finally, when carrying an object with a human with implicit communication, the human intent (as it relates to desired motion) must be modeled. With commercially available sensors and open-source software it is relatively straightforward to track a humans head pose, feet motion, torso and the applied wrench through the manipulator. If desired, these can be combined with predefined gestures that indicate desired motion, or the relation between these quantities and the intended motion can be adapted from a model prior or learned with no or few assumptions using deep learning methods [20, 67]. Studies show that human collaborators prefer when robotic teammates can anticipate their intent as this makes the interaction feel more natural [22]. In implementation, this usually takes the form of the robot being proactive, by using intent estimation as a feed-forward term in control [6, 27]. Another challenging task is autonomous pouring, where similar to cooperative transport the objective for successful task completion is clear however the pouring model may be unknown.

2.4 Autonomous Pouring

Robotic autonomous pouring is an important technology for improving the safety, productivity and repeatability in the wet laboratory setting as well as increasing the abilities of mobile manipulators in domestic environments. Work has been done to allow robots to effective manipulate containers and pour liquids in kitchen settings [8, 51, 53]. To effectively handle fluids, the liquid must first be observable. Many methods have been used to track fluids, from tracking the water during flight using thermal imaging or optical flow [61, 80] to detecting the surface of standing fluid [56] or in a clear pouring container classifying the amount of fluid into discrete amounts [45]. Once the fluid has been observed, the pouring process must be modeled. This can be done by either designing pouring dynamics that resembles the observed physics then using system identification to improve estimated parameters [49, 72], or by learning effective motion primitives that produce the expected flowrate correlated with joints states [34, 58, 69] or container geometry [11, 38].

In order to pour precisely, the robot must perform feedback control to pour the desired amount of fluid. To pour relatively quickly, smooth pouring trajectories can be designed with input shapers to transport the filled container to the pour location and pour without sloshing [2, 4, 85], and it is noted in [33] that minimum jerk motions are especially effective in minimizing sloshing. Depending on the desired degree of precision, the volume of the fluid can be generally classified to set volume amounts [62], or the fluid can be directly observed for more precision [28]. The geometry of the pouring container determines the relation between the robots angular motion and the pouring rate. The geometry of the container and flowrate can be explicitly modeled [28, 50] or the container model can be abstracted by realizing that pour motions for full containers if followed are successful for any starting volume and describes a maximum volume versus angle curve for sufficiently slow and steady pours [42].

Chapter 3

Mathematical Preliminaries

In this section the mathematical foundation is presented for controlling manipulators for complex tasks in unstructured environments. First, the essentials of mobile manipulator kinematics and dynamics is presented. Then the fundamentals of optimization and optimal control is discussed, which then sets the stage for model predictive control. In real world scenarios, the estimated task models may need to be adjusted and adapted as more information is obtained. This leads to a discussion of state estimation, then model approximation and adaptive control. When the main structure of the model is unknown, learning techniques can be leveraged to construct a model from data. Finally, given an adequate model representation, the essentials for nonlinear control and stability are presented. While most of the work in later chapters require model estimation in addition to nonlinear control, the discussion of considerations leading to these methods is valuable when considering deployable robots.

3.1 Mobile Manipulator Kinematics

Consider a rigid body, the degree of freedom (DOF) of the body describes the number of parameters (or coordinates) required to fully describe the pose of the rigid body. Holonomic constraints are those that are functions of the coordinate states. If these constraints are not explicit functions of time they are called scleronomic constraints. Whereas if they are they are functions of time, they are referred to as rheonomous constraints. A constraint that is a function of the derivative of the coordinates is called non-holonomic. Generalized coordinates q_i are defined as those that are independent of each other and are not subject to holonomic constraints, and therefore describe the effective degree of freedom for the system. If the constrained system is in equilibrium in the presence of external forces then the principle of virtual work states "The work done by external forces corresponding to any set of virtual displacements is zero." [65]. For a system that is not in static equilibrium, D'Alembert's principle considers the balance of external forces and the use of the differentiated momentum of a particle as a force acting over a 'virtual' (infinitesimal) displacement. This paves the way for solving the equations of motion of a system as a function of the kinetic and potential energy as well as external forces, and is formally referred to as the Euler-Lagrange equations of motion. When ordinary coordinates are used, the constraints restrict motion. These can be represented in the dynamics using the representation of Lagrange multipliers that essentially prescribe the required force to ensure the stated constraint is respected during motion.

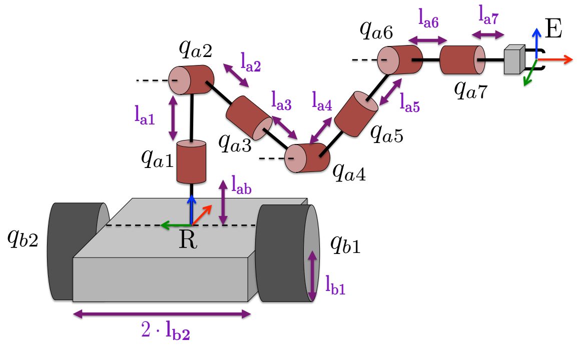


Figure 3.1: Mobile Redundant Manipulator Diagram: Frames R, E are robot and end-effector frames respectively. Angular coordinates of manipulator and base represented with q_i and the generalized coordinates of the base is the SE(2) constrained location of frame R and the joints $q_{\mathrm{A},i}$ for the manipulator.

First, define the pose of a rigid body denoted in the frame A with respect to the inertial or world frame W as

$$\boldsymbol{p}_{\mathrm{WA}} = \begin{bmatrix} x & y & z & \phi_x & \phi_y & \phi_z \end{bmatrix}^T \tag{3.1}$$

where x, y, z are the translation components and $\boldsymbol{\phi}$ is the vector of exponential coordinates where the equivalent rotation matrix can be found using the hat (skew-symmetric matrix) and exponential maps: $R = \exp(\boldsymbol{\phi}^{[\times]})$. The $T(\,\cdot\,)$ operator is defined to return the frame transformation matrix given the pose:

$$T(\boldsymbol{p}_{\mathrm{WA}}) = T_{\mathrm{WA}} = \begin{bmatrix} R_{\mathrm{WA}} & \boldsymbol{d}_{\mathrm{WA}} \\ \mathbf{0} & 1 \end{bmatrix} \tag{3.2}$$

where $\boldsymbol{d}_{\mathrm{WA}} = \begin{bmatrix} x & y & z \end{bmatrix}^T$ is the translation component of $\boldsymbol{p}_{\mathrm{WA}}$ and is the location of the origin of frame A in W.

Figure 3.1 presents the general mobile manipulator in terms of the reference frames R, angular coordinates q and relevant length quantities l_i. In this thesis, the frame R is the robot frame and sits on the mobile base and the z-axis is the same axis as first joint of the manipulator $q_{\mathrm{A},1}$. The generalized coordinates for the system are the manipulator angles q_{A}, and the SE(2) location of frame R: (x, y, ϕ_z) as the base has three degrees of freedom and no additional holonomic constraints in SE(2).

3.1.1 Mobile Base Kinematics

The constraints for the base are the non-holonomic 'knife-edge' constraint which restricts motion along axis that passes through both wheels (y-axis of frame R in Figure 3.1) as well as a no slip constraint while the wheels move. The first constraint is expressed by:

$$\begin{bmatrix} -sin(\phi_z) & cos(\phi_z) & 0 & 0 & 0 & 0 \end{bmatrix} \dot{\boldsymbol{p}}_{\mathrm{WR}} = 0 \tag{3.3}$$

The constraint that ensures the wheels do not slip can be expressed as:

$$\begin{bmatrix} -cos(\phi_z) & -sin(\phi_z) & 0 & 0 & 0 & 0 & l_{b1} & l_{b1} \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{p}}_{\mathrm{WR}} \\ \dot{q}_{b,2} \\ \dot{q}_{b,1} \end{bmatrix} = 0 \tag{3.4}$$

As the motion of the mobile is restricted to SE(2), the kinematics of the system is described by

$$\dot{\boldsymbol{p}}_{\mathrm{WR}} = \begin{bmatrix} l_{b1}cos(\phi_z) & l_{b1}sin(\phi_z) & 0 & 0 & 0 & -\frac{l_{b1}}{l_{b2}} \\ l_{b1}cos(\phi_z) & l_{b1}sin(\phi_z) & 0 & 0 & 0 & \frac{l_{b1}}{l_{b2}} \end{bmatrix}^T \begin{bmatrix} \dot{q}_{b2} \\ \dot{q}_{b1} \end{bmatrix} \tag{3.5}$$

where l_{b1} is the radius of the base wheel and $2 \cdot l_{b2}$ is the width of the base platform as shown in Figure 3.1.

3.1.2 Manipulator Kinematics

The manipulator has kinematic redundancy if the number of joints is greater than that needed to execute a task. In manipulation the task is usually associated with the placement of the end-effector which is described by 6-DOF pose vector as in (3.1).

The pose of the end-effector is a function of the joint angles and linkage lengths. By defining a frame associated with every joint, the frames can be composed to find the transform between any set of frames in the manipulator (here between frames i and $i + n$)

$$T_{i+n,i} = T_{i+n,i+(n-1)} \cdots T_{i+2,i+1} T_{i+1,i}. \tag{3.6}$$

Hence, every frame is a function of the current joint angle and all preceding joint angles. A manipulators forward kinematics transformation matrix can be represented by an equivalent and unique Danavit-Hartenberg (DH) convention by a selection of frames that satisfy the two constraints that each consecutive DH frame's x-axis be perpendicular to and intersect the previous frames z-axis, where the respective joint angle rotation is about each frames z-axis. This is particularly useful in practical scenarios when the kinematics of a manipulator must be identified through measurements due to unknown true dimensions of the linkages [65].

The pose of the end-effector connected with a base in frame R represented in the world (inertial reference) frame W is described by forward kinematic function $FK(\cdot)$

$$\boldsymbol{p}_{\mathrm{WE}} = T_{\mathrm{WE}} = FK(\boldsymbol{q}_{\mathrm{A}}, \boldsymbol{p}_{\mathrm{WR}}) \tag{3.7}$$

Differentiating and denoting the generalized coordinates for the base (which describe the location in SE(3)) as $\boldsymbol{p}_{\mathrm{WR}}$:

$$\begin{aligned}\dot{\boldsymbol{p}}_{\text{WE}} &= \sum_i \frac{\partial FK(\boldsymbol{q}_{\text{A}}, \boldsymbol{p}_{\text{WR}})}{\partial q_i} \frac{dq_i}{dt} \\ &= J(\boldsymbol{q}_{\text{A}}, \boldsymbol{p}_{\text{WR}}) \begin{bmatrix} \dot{\boldsymbol{q}}_{\text{A}} \\ \dot{\boldsymbol{q}}_{\text{B}} \end{bmatrix} \end{aligned} \tag{3.8}$$

Where J is called the Jacobian. Note that this Jacobian can be subdivided into the following components where the subscripts v and ω denote linear and angular velocity and subscripts a and b denote the arm and base respectively:

$$J(\boldsymbol{q}_{\text{A}}, \boldsymbol{p}_{\text{WR}}) = \begin{bmatrix} J_{v\text{A}} & J_{v\text{B}} \\ J_{\omega\text{A}} & J_{\omega\text{B}} \end{bmatrix}. \tag{3.9}$$

The Jacobian can be solved for any rigid body in the system, and provides the twist of the rigid body in the inertial frame W. Therefore throughout the rest of the thesis it can be assumed that the Jacobian provides the twist in the inertial frame. The components $J_{v\text{B}}, J_{\omega\text{B}}$ come from the matrix in (3.5).

An external wrench $\boldsymbol{w}$ (force and torque) applied at a point on the manipulator with the associated Jacobian J in equilibrium exerts a torque in the joints by D'Alembert's principle of virtual work $\delta\mathcal{W}$ as

$$\begin{aligned} \delta\mathcal{W} &= \boldsymbol{\tau}^T \delta\boldsymbol{q}_{\text{A}} - \boldsymbol{w}^T \delta\boldsymbol{p}_{\text{WE}} \\ \boldsymbol{\tau}^T \delta\boldsymbol{q}_{\text{A}} &= \boldsymbol{w}^T \delta\boldsymbol{p}_{\text{WE}} \\ \boldsymbol{\tau}^T \delta\boldsymbol{q}_{\text{A}} &= \boldsymbol{w}^T J \delta\boldsymbol{q}_{\text{A}} \\ \boldsymbol{\tau} &= J^T \boldsymbol{w} \end{aligned} \tag{3.10}$$

3.2 Mobile Manipulator Dynamics

The Euler-Lagrange equations of motion can be directly derived from D'Alembert's principle with the kinetic energy terms being derived from the fictitious additional force (derivative of system momentum) multiplied by the virtual displacement and the external forces accounting for input forces, potential fields (such as the effect of gravity) and constraint forces [65]. The kinetic energy of a rigid body is defined as

$$\mathcal{K} = \frac{1}{2}m\boldsymbol{v}^2 + \frac{1}{2}\boldsymbol{\omega}\mathcal{I}\boldsymbol{\omega} \tag{3.11}$$

where $\boldsymbol{v}$, $\boldsymbol{\omega}$ are linear and angular velocities respectively, and m, $\mathcal{I}$ are the mass and inertia tensor of the rigid body respectively. If the potential energy of the system is described by $\mathcal{P}(\boldsymbol{r})$, where $\boldsymbol{r}$ here is the coordinates of the system, then the Lagrangian is defined as

$$\mathcal{L} = \mathcal{K} - \mathcal{P} \tag{3.12}$$

Given default coordinates $(\boldsymbol{r})$ that are a function of the generalized coordinates $\boldsymbol{r}_j = \boldsymbol{r}_j(q_1, \ldots, q_n)$ $j = 1, \ldots, m$, the set of C_h holonomic constraint equations is defined as

$$h_i(\boldsymbol{r}, t) = 0 \;\; \forall i \in [1, C_h] \tag{3.13}$$

and a set of C_{nh} non-holonomic constraint equations is defined as

$$n_k(\dot{\boldsymbol{r}}, t) = 0 \;\; \forall k \in [1, C_{nh}]. \tag{3.14}$$

The Lagrange equations of motion of the first kind are with respect to the default coordinates are

$$\frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{r}_j} - \frac{\partial \mathcal{L}}{\partial r_j} = \tau_j + \sum_{i=1}^{C_h} \lambda_i \frac{\partial h_i}{\partial r_j} + \sum_{k=1}^{C_{nh}} \lambda_k \frac{\partial n_k}{\partial \dot{r}_j} \tag{3.15}$$

where τ_j is the externally applied force that is not subject to constraints and results in motion of the system and λ are Lagrange multipliers that embody the environment force required to maintain the given constraint. The generalized coordinates are defined as the minimum number of independent variables required to express the pose of a body, therefore the selection of these coordinates are chosen such that they do not violate holonomic constraints. However, these generalized coordinates may be subjected to non-holonomic constraints

$$a_k(\dot{\boldsymbol{q}}, t) = 0 \;\; \forall k \in [1, C_{nh}]. \tag{3.16}$$

This is the basis of Lagrange equations of motion of the second kind, expressed in terms of N generalized coordinates q_i

$$\frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{q}_i} - \frac{\partial \mathcal{L}}{\partial q_i} = \tau_i + \sum_{k=1}^{C_{nh}} \lambda_k \frac{\partial a_k}{\partial \dot{q}_i} \;\; \forall i \in [1, N] \tag{3.17}$$

3.2.1 Manipulator Dynamics

Consider first a static base, and therefore the dynamics of just a manipulator. Two common approaches are the Newton-Euler formulation and the Euler-Lagrange equations of motion. For real-time systems with no ‘loops’ in the linkages (with loops defined as the ability to return to a linkage frame without revisiting any other frame) the Newton-Euler method is much faster to calculate. The Newton-Euler method is recursive in that the inertial effects on all of the linkages are found by applying Newtons third law at linkage interfaces while considering the rigid body dynamics of the individual linkages. The parameters of interest are the accelerations of the center of mass and the end (at the next joint location) of the link, the angular velocity and acceleration of the link frame in the inertial frame. In addition, the gravity force vector and relative frame rotation matrices must be known. Additionally, the forces and torque applied on or by the links must be also found. Starting at the base of the manipulator with known angular and linear velocities and accelerations (constant for static base), the kinematic equations for the manipulator along with observable joint position, velocity and acceleration are used to find the consecutive angular velocity, acceleration and linear accelerations of the links all the way to the end-effector link frame of the manipulator. Given these linear and angular accelerations and velocities with known wrench applied and the end of the manipulator the force, torque equations can be solved consecutively from the end of the manipulator to the base of the manipulator [65].

The linear velocity of each joint is described by

$$\boldsymbol{v}_{\mathrm{A}_i} = J_{v\mathrm{A}_i,ci}\dot{\boldsymbol{q}}_{\mathrm{A}} \tag{3.18}$$

where $J_{v\mathrm{A}_i,ci}$ is the Jacobian for the linear motion of the centroid of the $i'th$ link. And angular velocity is described by

$$\boldsymbol{\omega}_{\mathrm{A}_i} = J_{\omega\mathrm{A}_i,ci}\dot{\boldsymbol{q}}_{\mathrm{A}} \tag{3.19}$$

where $J_{\omega\mathrm{A}_i,ci}$ is the Jacobian for the angular motion of the centroid of the $i'th$ link. The kinetic energy of the manipulator with n joints is then

$$\mathcal{K}_{\mathrm{A}} = \sum_i^n \left(\frac{1}{2}m_{\mathrm{A}_i}\boldsymbol{v}_{\mathrm{A}_i}^T\boldsymbol{v}_{\mathrm{A}_i} + \frac{1}{2}\boldsymbol{\omega}_{\mathrm{A}_i}^T\mathcal{I}'_{\mathrm{A}_i}\boldsymbol{\omega}_{\mathrm{A}_i}\right) \tag{3.20}$$

Where $\mathcal{I}'_{\mathrm{A}_i}$ is the inertia tensor for the $i'th$ link in the inertial frame using the similarity transformation

$$\mathcal{I}'_{\mathrm{A}_i} = R_{\mathrm{WA}_i}\mathcal{I}_{\mathrm{A}_i}R_{\mathrm{WA}_i}^T \tag{3.21}$$

and $\mathcal{I}_{\mathrm{A}_i}$ is the inertial frame in the $i'th$ link frame. Expanding produces

$$\begin{aligned}\mathcal{K}_{\mathrm{A}} &= \sum_i^n \frac{1}{2}\dot{q}_{\mathrm{A}_i}^T\left(m_{\mathrm{A}_i}J_{v\mathrm{A}_i,ci}^T J_{v\mathrm{A}_i,ci} + J_{\omega\mathrm{A}_i,ci}^T R_{\mathrm{WA}_i}\mathcal{I}_{\mathrm{A}_i}R_{\mathrm{WA}_i}^T J_{\omega\mathrm{A}_i,ci}\right)\dot{q}_{\mathrm{A}_i} && (3.22)\\ &= \frac{1}{2}\dot{\boldsymbol{q}}_{\mathrm{A}}^T D_{\mathrm{A}}(q)\dot{\boldsymbol{q}}_{\mathrm{A}} && (3.23)\end{aligned}$$

Where $D_{\mathrm{A}}(q)$ is a $n \times n$ matrix called the inertia matrix for the manipulator. The potential function for the manipulator is described by

$$\mathcal{P}_{\mathrm{A}} = \sum_i^n m_{\mathrm{A}_i}\boldsymbol{g}^T\boldsymbol{r}_{ci} \tag{3.24}$$

where $\boldsymbol{g}$ is the gravity vector, and $\boldsymbol{r}_{ci}$ is the a vector in the inertial frame from frame R to the centroid of the $i'th$ linkage. Then defining the entries of the inertia matrix $D_{\mathrm{A}}(q)$ as $d_{\mathrm{A},ij}(q)$ the Lagrangian becomes

$$\mathcal{L} = \mathcal{K}_{\mathrm{A}} - \mathcal{P}_{\mathrm{A}} = \frac{1}{2}\sum_{i}^{n} d_{\mathrm{A},ij}(q)\dot{q}_i\dot{q}_j - \mathcal{P}_{\mathrm{A}}(q) \tag{3.25}$$

Then the equations of motion including force due to gravity, the applied torques τ_i applied each joint become for each joint

$$\begin{align}
\tau_k &= \frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{q}_k} - \frac{\partial \mathcal{L}}{\partial q_k} \tag{3.26}\\
&= \left(\sum_j d_{kj}\ddot{q}_j + \sum_{i,j}\left(\frac{\partial d_{kj}}{\partial q_i}\dot{q}_i\right)\dot{q}_j\right) - \left(\frac{1}{2}\sum_{i,j}\frac{\partial d_{ij}}{\partial q_k}\dot{q}_i\dot{q}_j - \frac{\partial \mathcal{P}}{\partial q_k}\right) \tag{3.27}\\
&= \sum_j d_{kj}\ddot{q}_j + \sum_{i,j}\left(\frac{\partial d_{kj}}{\partial q_i} - \frac{1}{2}\frac{\partial d_{ij}}{\partial q_k}\right)\dot{q}_i\dot{q}_j + \frac{\partial \mathcal{P}}{\partial q_k} \tag{3.28}\\
&= \sum_j d_{kj}\ddot{q}_j + \sum_{i,j}\frac{1}{2}\left(\frac{\partial d_{kj}}{\partial q_i} + \frac{\partial d_{ki}}{\partial q_j} - \frac{\partial d_{ij}}{\partial q_k}\right)\dot{q}_i\dot{q}_j + \frac{\partial \mathcal{P}}{\partial q_k} \tag{3.29}\\
&= \sum_j d_{kj}(q)\ddot{q}_j + \sum_{i,j}\frac{1}{2}c_{ijk}(q)\dot{q}_i\dot{q}_j + g_k(q) \tag{3.30}
\end{align}$$

The term $g_k(q)$ denotes the potential energy. Note that the expansion in (3.29) comes from the property that the inertia matrix is symmetric therefore

$$\frac{\partial d_{kj}}{\partial q_i} = \frac{\partial d_{ki}}{\partial q_j} \tag{3.31}$$

and c_{ijk} is called the Christoffel symbols of the first kind and are the Coriolis and centrifugal forces on the manipulator [65]. It's also worth noting that for the manipulator there is an alternate representation leveraging the linearity of the manipulator model property with respect to suitable dynamic parameters. This is done by realizing that the Lagrangian in (3.25) with each links mass as m_i, position between the links rotor and center of mass as $\boldsymbol{r}_{ci}$, and inertia tensor at the rotor $\tilde{\mathcal{I}}_i = \mathcal{I}_i + m_i\boldsymbol{r}_{ci}^{[\times]T}\boldsymbol{r}_{ci}^{[\times]}$ from Huygens-Steiner theorem, can be expressed as

$$\mathcal{L} = \sum_{i}^{n} \left(\boldsymbol{\beta}_{\mathcal{K},i}^{T} - \boldsymbol{\beta}_{\mathcal{P},i}^{T} \right) \boldsymbol{\pi}_i \tag{3.32}$$

where the terms $\boldsymbol{\beta}_{\mathcal{K},i}^{T}$, $\boldsymbol{\beta}_{\mathcal{P},i}^{T}$ are 11×1 vectors that group the kinetic and potential terms to be multiplied with the vector $\boldsymbol{\pi}_i$, which is the dynamic parameters for the $i'th$ link containing the mass, the first moment of inertia and the six elements of the inertia tensor

$$\begin{aligned} \boldsymbol{\pi}_i = \Big[m_i \quad & m_i \boldsymbol{r}_{ci,x} \quad m_i \boldsymbol{r}_{ci,y} \quad m_i \boldsymbol{r}_{ci,z} \quad \ldots \\ & \tilde{\mathcal{I}}_{i,xx} \quad \tilde{\mathcal{I}}_{i,xy} \quad \tilde{\mathcal{I}}_{i,xz} \quad \tilde{\mathcal{I}}_{i,yy} \quad \tilde{\mathcal{I}}_{i,yz} \quad \tilde{\mathcal{I}}_{i,zz} \quad \tilde{\mathcal{I}}_{m_i} \Big]^T \end{aligned} \tag{3.33}$$

where the last term $\tilde{\mathcal{I}}_{m_i}$ is the motor inertia if desired. Then solving for the equations of motion as in (3.26) this becomes

$$\tau_k = \sum_{j=1}^{n} \boldsymbol{y}_{kj}^{T} \boldsymbol{\pi}_j \tag{3.34}$$

where

$$\boldsymbol{y}_{kj} = \frac{d}{dt} \frac{\partial \boldsymbol{\beta}_{\mathcal{K},j}}{\partial \dot{q}_k} - \frac{\partial \boldsymbol{\beta}_{\mathcal{K},j}}{\partial q_i} + \frac{\partial \boldsymbol{\beta}_{\mathcal{P},k}}{\partial q_i} \tag{3.35}$$

which allows (3.34) to be expressed as

$$\boldsymbol{\tau} = Y(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}) \boldsymbol{\pi} \tag{3.36}$$

where $Y(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}})$ is an upper triangular matrix as shown in [64].

3.2.2 Base Dynamics

For the base dynamics alone, assume that the joints for the manipulator are locked in a given configuration such that the vector $\boldsymbol{q}_{\mathrm{A}}$ is a constant vector. However, the mass and inertia of the manipulator mounted on the base cannot be ignored. By defining the $\boldsymbol{r}_{ci}$ as the vector from the centroid of the $i'th$ linkage to the R frame, the effective inertia at frame R of each joint in the joint frame is found using a generalization of the Huygens–Steiner (parallel axis) theorem. Given that the inertia tensor is found by integrating over the volume of a rigid body there is the relation $dm = \rho(x, y, z)dV$ that relates differential mass and volume with a function of density.

$$\tilde{\mathcal{I}}_{\mathrm{A}_i,xx} = \int \left((y+\boldsymbol{r}_{ci,y})^2 + (z+\boldsymbol{r}_{ci,z})^2\right) dm \tag{3.37}$$

$$= \int \left((y^2+z^2) + 2\boldsymbol{r}_{ci,y}y + 2\boldsymbol{r}_{ci,z}z + \boldsymbol{r}_{ci,y}^2 + \boldsymbol{r}_{ci,z}^2\right) dm \tag{3.38}$$

$$= \int (y^2+z^2)dm + (\boldsymbol{r}_{ci,y}^2 + \boldsymbol{r}_{ci,z}^2) \int dm \tag{3.39}$$

$$= \mathcal{I}_{\mathrm{A}_i,xx} + (\boldsymbol{r}_{ci,y}^2 + \boldsymbol{r}_{ci,z}^2) \int dm \tag{3.40}$$

Note that $2\boldsymbol{r}_{ci,y}\int ydm = 2\boldsymbol{r}_{ci,z}\int zdm = 0$ since this integration is centered about the center of mass. Therefore similarly for all inertia tensor entries

$$\tilde{\mathcal{I}}_{\mathrm{A}_i,xx} = \mathcal{I}_{\mathrm{A}_i,xx} + (\boldsymbol{r}_{ci,y}^2 + \boldsymbol{r}_{ci,z}^2)m_{\mathrm{A}_i} \tag{3.41}$$

$$\tilde{\mathcal{I}}_{\mathrm{A}_i,yy} = \mathcal{I}_{\mathrm{A}_i,yy} + (\boldsymbol{r}_{ci,x}^2 + \boldsymbol{r}_{ci,z}^2)m_{\mathrm{A}_i} \tag{3.42}$$

$$\tilde{\mathcal{I}}_{\mathrm{A}_i,zz} = \mathcal{I}_{\mathrm{A}_i,zz} + (\boldsymbol{r}_{ci,x}^2 + \boldsymbol{r}_{ci,y}^2)m_{\mathrm{A}_i} \tag{3.43}$$

$$\tilde{\mathcal{I}}_{\mathrm{A}_i,xy} = \tilde{\mathcal{I}}_{\mathrm{A}_i,yx} = \mathcal{I}_{\mathrm{A}_i,xy} - \boldsymbol{r}_{ci,x}\boldsymbol{r}_{ci,y}m_{\mathrm{A}_i} \tag{3.44}$$

$$\tilde{\mathcal{I}}_{\mathrm{A}_i,xz} = \tilde{\mathcal{I}}_{\mathrm{A}_i,zx} = \mathcal{I}_{\mathrm{A}_i,xz} - \boldsymbol{r}_{ci,x}\boldsymbol{r}_{ci,z}m_{\mathrm{A}_i} \tag{3.45}$$

$$\tilde{\mathcal{I}}_{\mathrm{A}_i,yz} = \tilde{\mathcal{I}}_{\mathrm{A}_i,zy} = \mathcal{I}_{\mathrm{A}_i,yz} - \boldsymbol{r}_{ci,y}\boldsymbol{r}_{ci,z}m_{\mathrm{A}_i} \tag{3.46}$$

$$\tag{3.47}$$

Hence

$$\tilde{\mathcal{I}}_{\mathrm{A}_i} = \mathcal{I}_{\mathrm{A}_i} + m_{\mathrm{A}_i}\begin{bmatrix} (\boldsymbol{r}_{ci,y}^2 + \boldsymbol{r}_{ci,z}^2) & -\boldsymbol{r}_{ci,x}\boldsymbol{r}_{ci,y} & -\boldsymbol{r}_{ci,x}\boldsymbol{r}_{ci,z} \\ -\boldsymbol{r}_{ci,x}\boldsymbol{r}_{ci,y} & (\boldsymbol{r}_{ci,x}^2 + \boldsymbol{r}_{ci,z}^2) & -\boldsymbol{r}_{ci,y}\boldsymbol{r}_{ci,z} \\ -\boldsymbol{r}_{ci,x}\boldsymbol{r}_{ci,z} & -\boldsymbol{r}_{ci,y}\boldsymbol{r}_{ci,z} & (\boldsymbol{r}_{ci,x}^2 + \boldsymbol{r}_{ci,y}^2) \end{bmatrix} \tag{3.48}$$

$$= \mathcal{I}_{\mathrm{A}_i} + m_{\mathrm{A}_i}\boldsymbol{r}_{ci}^{[\times]T}\boldsymbol{r}_{ci}^{[\times]} \tag{3.49}$$

The base kinetic energy is

$$\mathcal{K}_{\mathrm{B}} = \frac{1}{2} m_{\mathrm{B}} \boldsymbol{v}_{\mathrm{B}}^T \boldsymbol{v}_{\mathrm{B}} + \frac{1}{2} \boldsymbol{\omega}_{\mathrm{B}}^T \mathcal{I}_{\mathrm{B}}' \boldsymbol{\omega}_{\mathrm{B}} \tag{3.50}$$

The augmented mass and inertia are

$$\tilde{m}_{\mathrm{B}} = m_{\mathrm{B}} + \sum_i^n m_{\mathrm{A}_i} \tag{3.51}$$

$$\tilde{\mathcal{I}}_{\mathrm{B}} = R_{\mathrm{WR}} \mathcal{I}_{\mathrm{B}} R_{\mathrm{WR}}^T + \sum_i^n R_{\mathrm{WA}_i} \left(\mathcal{I}_{\mathrm{A}_i} + m_{\mathrm{A}_i} \boldsymbol{r}_{ci}^{[\times]T} \boldsymbol{r}_{ci}^{[\times]} \right) R_{\mathrm{WA}_i}^T \tag{3.52}$$

where these again are static quantities as the manipulator joints are considered locked. For base wheel velocities $\dot{q}_{b,1}, \dot{q}_{b,2}$ which are the right and left wheels respectively, the velocity of the system is described in terms of these by the Jacobian components

$$\begin{aligned} \boldsymbol{v}_{\mathrm{B}} &= \begin{bmatrix} cos(\phi_{R,z}) & sin(\phi_{R,z}) & 0 \\ cos(\phi_{R,z}) & sin(\phi_{R,z}) & 0 \end{bmatrix}^T \begin{bmatrix} \dot{q}_{b,1} \\ \dot{q}_{b,2} \end{bmatrix} \\ &= J_{v\mathrm{B}} \begin{bmatrix} \dot{q}_{b,1} \\ \dot{q}_{b,2} \end{bmatrix} \end{aligned} \tag{3.53}$$

$$\begin{aligned} \boldsymbol{\omega}_{\mathrm{B}} &= \begin{bmatrix} 0 & 0 & \frac{l_{b1}}{l_{b2}} \\ 0 & 0 & -\frac{l_{b1}}{l_{b2}} \end{bmatrix}^T \begin{bmatrix} \dot{q}_{b,1} \\ \dot{q}_{b,2} \end{bmatrix} \\ &= J_{\omega\mathrm{B}} \begin{bmatrix} \dot{q}_{b,1} \\ \dot{q}_{b,2} \end{bmatrix} \end{aligned} \tag{3.54}$$

as in (3.5). Then by substituting (3.53) and (3.54) along with (3.51) and (3.52) into (3.50) and realizing that as the system is confined to SE(2) the potential function $\mathcal{P}(q) = 0$, the Lagrangian (3.12) can be solved with torque inputs to the wheels τ_k

$$\tau_k = \frac{d}{dt} \frac{\partial \mathcal{L}}{\partial \dot{q}_{b,k}} \tag{3.55}$$

3.2.3 Coupled Dynamics for the Manipulator and Base

When both the base and the manipulator are allowed to move, the coupled dynamics must be considered. This can again be solved using Euler-Lagrange equations of motion. First the velocity of each rigid body in the system must be found with respect to the inertial frame W. For the base rigid body, let the velocity be described by (3.53) and (3.54). For each manipulator linkage, let the linear and angular velocities be described in the inertial frame by

$$\boldsymbol{v}_{\mathrm{A}_i} = \boldsymbol{v}_{\mathrm{B}} + \boldsymbol{\omega}_{\mathrm{B}}^{[\times]} \boldsymbol{r}_{ci} + J_{v\mathrm{A}_i,ci} \dot{\boldsymbol{q}}_{\mathrm{A}} \tag{3.56}$$

$$\boldsymbol{\omega}_{\mathrm{A}_i} = J_{\omega\mathrm{A}_i,ci} \dot{\boldsymbol{q}}_{\mathrm{A}} + \boldsymbol{\omega}_{\mathrm{B}} \tag{3.57}$$

Therefore the total kinetic energy for the base and manipulator is

$$\mathcal{K}_{\mathrm{AB}} = \frac{1}{2}\left(m_{\mathrm{B}} \boldsymbol{v}_{\mathrm{B}}^T \boldsymbol{v}_{\mathrm{B}} + \boldsymbol{\omega}_{\mathrm{B}}^T \mathcal{I}'_{\mathrm{B}} \boldsymbol{\omega}_{\mathrm{B}} + \sum_i^n \left(m_{\mathrm{A}_i} \boldsymbol{v}_{\mathrm{A}_i}^T \boldsymbol{v}_{\mathrm{A}_i} + \boldsymbol{\omega}_{\mathrm{A}_i}^T \mathcal{I}'_{\mathrm{A}_i} \boldsymbol{\omega}_{\mathrm{A}_i} \right) \right) \tag{3.58}$$

And the potential energy is just the portion from the manipulator in (3.24). Expanding $m_{\mathrm{B}} \boldsymbol{v}_{\mathrm{B}}^T \boldsymbol{v}_{\mathrm{B}}$ and $\boldsymbol{\omega}_{\mathrm{B}}^T \mathcal{I}'_{\mathrm{B}} \boldsymbol{\omega}_{\mathrm{B}}$:

$$m_{\mathrm{B}} \boldsymbol{v}_{\mathrm{B}}^T \boldsymbol{v}_{\mathrm{B}} = m_{\mathrm{B}} \dot{\boldsymbol{q}}_{\mathrm{B}}^T J_{v\mathrm{B}}^T J_{v\mathrm{B}} \dot{\boldsymbol{q}}_{\mathrm{B}} \tag{3.59}$$

$$\begin{aligned} \boldsymbol{\omega}_{\mathrm{B}}^T \mathcal{I}'_{\mathrm{B}} \boldsymbol{\omega}_{\mathrm{B}} &= \boldsymbol{\omega}_{\mathrm{B}}^T R_{\mathrm{WR}} \mathcal{I}_{\mathrm{B}} R_{\mathrm{WR}}^T \boldsymbol{\omega}_{\mathrm{B}} \\ &= \dot{\boldsymbol{q}}_{\mathrm{B}}^T J_{\omega\mathrm{B}}^T R_{\mathrm{WR}} \mathcal{I}_{\mathrm{B}} R_{\mathrm{WR}}^T J_{\omega\mathrm{B}} \dot{\boldsymbol{q}}_{\mathrm{B}} \end{aligned} \tag{3.60}$$

Expanding the $m_{\mathrm{A}_i} \boldsymbol{v}_{\mathrm{A}_i}^T \boldsymbol{v}_{\mathrm{A}_i}$ terms using (3.56), produces

$$\begin{aligned}
m_{\mathrm{A}_i}\boldsymbol{v}_{\mathrm{A}_i}^T\boldsymbol{v}_{\mathrm{A}_i} &= m_{\mathrm{A}_i}\left(\boldsymbol{v}_\mathrm{B} + \boldsymbol{\omega}_\mathrm{B}^{[\times]}\boldsymbol{r}_{ci} + J_{v\mathrm{A}_i,ci}\dot{\boldsymbol{q}}_\mathrm{A}\right)^T\left(\boldsymbol{v}_\mathrm{B} + \boldsymbol{\omega}_\mathrm{B}^{[\times]}\boldsymbol{r}_{ci} + J_{v\mathrm{A}_i,ci}\dot{\boldsymbol{q}}_\mathrm{A}\right) \\
&= m_{\mathrm{A}_i}\Big(\boldsymbol{v}_\mathrm{B}^T\boldsymbol{v}_\mathrm{B} + \boldsymbol{v}_\mathrm{B}^T\boldsymbol{\omega}_\mathrm{B}^{[\times]}\boldsymbol{r}_{ci} + \boldsymbol{v}_\mathrm{B}^T J_{v\mathrm{A}_i,ci}\dot{\boldsymbol{q}}_\mathrm{A} + (\boldsymbol{\omega}_\mathrm{B}^{[\times]}\boldsymbol{r}_{ci})^T\boldsymbol{v}_\mathrm{B} \\
&+ (\boldsymbol{\omega}_\mathrm{B}^{[\times]}\boldsymbol{r}_{ci})^T\boldsymbol{\omega}_\mathrm{B}^{[\times]}\boldsymbol{r}_{ci} + (\boldsymbol{\omega}_\mathrm{B}^{[\times]}\boldsymbol{r}_{ci})^T J_{v\mathrm{A}_i,ci}\dot{\boldsymbol{q}}_\mathrm{A} + \dot{\boldsymbol{q}}_\mathrm{A}^T J_{v\mathrm{A}_i,ci}^T\boldsymbol{v}_\mathrm{B} \\
&+ \dot{\boldsymbol{q}}_\mathrm{A}^T J_{v\mathrm{A}_i,ci}^T\boldsymbol{\omega}_\mathrm{B}^{[\times]}\boldsymbol{r}_{ci} + \dot{\boldsymbol{q}}_\mathrm{A}^T J_{v\mathrm{A}_i,ci}^T J_{v\mathrm{A}_i,ci}\dot{\boldsymbol{q}}_\mathrm{A}\Big) \qquad (3.61)
\end{aligned}$$

Using the property that $\boldsymbol{\omega}_\mathrm{B}^{[\times]}\boldsymbol{r}_{ci} = -\boldsymbol{r}_{ci}^{[\times]}\boldsymbol{\omega}_\mathrm{B}$ along with $\dot{\boldsymbol{q}}_\mathrm{B} = \begin{bmatrix}\dot{q}_{b,1} & \dot{q}_{b,2}\end{bmatrix}^T$ and with $J_{v\mathrm{B}}, J_{\omega\mathrm{B}}$ from (3.53) and (3.54) then (3.61) can be expressed as

$$\begin{aligned}
m_{\mathrm{A}_i}\boldsymbol{v}_{\mathrm{A}_i}^T\boldsymbol{v}_{\mathrm{A}_i} &= m_{\mathrm{A}_i}\Big(\dot{\boldsymbol{q}}_\mathrm{B}^T J_{v\mathrm{B}}^T J_{v\mathrm{B}}\dot{\boldsymbol{q}}_\mathrm{B} - \dot{\boldsymbol{q}}_\mathrm{B}^T J_{v\mathrm{B}}^T\boldsymbol{r}_{ci}^{[\times]}J_{\omega\mathrm{B}}\dot{\boldsymbol{q}}_\mathrm{B} + \dot{\boldsymbol{q}}_\mathrm{B}^T J_{v\mathrm{B}}^T J_{v\mathrm{A}_i,ci}\dot{\boldsymbol{q}}_\mathrm{A} \\
&- \dot{\boldsymbol{q}}_\mathrm{B}^T J_{\omega\mathrm{B}}^T\boldsymbol{r}_{ci}^{[\times]T}J_{v\mathrm{B}}\dot{\boldsymbol{q}}_\mathrm{B} + \dot{\boldsymbol{q}}_\mathrm{B}^T J_{\omega\mathrm{B}}^T\boldsymbol{r}_{ci}^{[\times]T}\boldsymbol{r}_{ci}^{[\times]}J_{\omega\mathrm{B}}\dot{\boldsymbol{q}}_\mathrm{B} + \dot{\boldsymbol{q}}_\mathrm{B}^T J_{\omega\mathrm{B}}^T\boldsymbol{r}_{ci}^{[\times]T}J_{v\mathrm{A}_i,ci}\dot{\boldsymbol{q}}_\mathrm{A} \\
&+\dot{\boldsymbol{q}}_\mathrm{A}^T J_{v\mathrm{A}_i,ci}^T J_{v\mathrm{B}}\dot{\boldsymbol{q}}_\mathrm{B} - \dot{\boldsymbol{q}}_\mathrm{A}^T J_{v\mathrm{A}_i,ci}^T\boldsymbol{r}_{ci}^{[\times]}J_{\omega\mathrm{B}}\dot{\boldsymbol{q}}_\mathrm{B} + \dot{\boldsymbol{q}}_\mathrm{A}^T J_{v\mathrm{A}_i,ci}^T J_{v\mathrm{A}_i,ci}\dot{\boldsymbol{q}}_\mathrm{A}\Big) \qquad (3.62)
\end{aligned}$$

likewise for the terms $\boldsymbol{\omega}_{\mathrm{A}_i}^T\mathcal{I}'_{\mathrm{A}_i}\boldsymbol{\omega}_{\mathrm{A}_i}$ using (3.54), (3.21) and expanding

$$\begin{aligned}
\boldsymbol{\omega}_{\mathrm{A}_i}^T\mathcal{I}'_{\mathrm{A}_i}\boldsymbol{\omega}_{\mathrm{A}_i} &= \left(J_{\omega\mathrm{A}_i,ci}\dot{\boldsymbol{q}}_\mathrm{A} + \boldsymbol{\omega}_\mathrm{B}\right)\mathcal{I}'_{\mathrm{A}_i}\left(J_{\omega\mathrm{A}_i,ci}\dot{\boldsymbol{q}}_\mathrm{A} + \boldsymbol{\omega}_\mathrm{B}\right) \\
&= \dot{\boldsymbol{q}}_\mathrm{A}^T J_{\omega\mathrm{A}_i,ci}^T R_{\mathrm{WA}_i}\mathcal{I}_{\mathrm{A}_i}R_{\mathrm{WA}_i}^T J_{\omega\mathrm{A}_i,ci}\dot{\boldsymbol{q}}_\mathrm{A} \\
&+ \dot{\boldsymbol{q}}_\mathrm{B}^T J_{\omega\mathrm{B}}^T R_{\mathrm{WA}_i}\mathcal{I}_{\mathrm{A}_i}R_{\mathrm{WA}_i}^T J_{\omega\mathrm{B}}\dot{\boldsymbol{q}}_\mathrm{B} \qquad (3.63)
\end{aligned}$$

As the kinetic energy is defined in terms of the inertia matrix and generalized coordinate velocities

$$\mathcal{K}_\mathrm{AB} = \frac{1}{2}\begin{bmatrix}\dot{\boldsymbol{q}}_\mathrm{A}^T & \dot{\boldsymbol{q}}_\mathrm{B}^T\end{bmatrix} D_\mathrm{AB}\begin{bmatrix}\dot{\boldsymbol{q}}_\mathrm{A} \\ \dot{\boldsymbol{q}}_\mathrm{B}\end{bmatrix}, \qquad (3.64)$$

the task becomes representing the inertia tensor. For simplicity, the following terms are defined as

$$\begin{aligned}
d_1 &= m_{\mathrm{B}} J_{v\mathrm{B}}^T J_{v\mathrm{B}} \\
d_2 &= J_{\omega\mathrm{B}}^T R_{\mathrm{WR}} \mathcal{I}_{\mathrm{B}} R_{\mathrm{WR}}^T J_{\omega\mathrm{B}} \\
d_3 &= m_{\mathrm{A}_i} J_{v\mathrm{B}}^T J_{v\mathrm{B}} \\
d_4 &= -m_{\mathrm{A}_i} J_{v\mathrm{B}}^T \boldsymbol{r}_{ci}^{[\times]} J_{\omega\mathrm{B}} \\
d_5 &= m_{\mathrm{A}_i} J_{v\mathrm{B}}^T J_{v\mathrm{A}_i,ci} \\
d_6 &= -m_{\mathrm{A}_i} J_{\omega\mathrm{B}}^T \boldsymbol{r}_{ci}^{[\times]T} J_{v\mathrm{B}} \\
d_7 &= m_{\mathrm{A}_i} J_{\omega\mathrm{B}}^T \boldsymbol{r}_{ci}^{[\times]T} \boldsymbol{r}_{ci}^{[\times]} J_{\omega\mathrm{B}} \\
d_8 &= m_{\mathrm{A}_i} J_{\omega\mathrm{B}}^T \boldsymbol{r}_{ci}^{[\times]T} J_{v\mathrm{A}_i,ci} \\
d_9 &= m_{\mathrm{A}_i} J_{v\mathrm{A}_i,ci}^T J_{v\mathrm{B}} \\
d_{10} &= -m_{\mathrm{A}_i} J_{v\mathrm{A}_i,ci}^T \boldsymbol{r}_{ci}^{[\times]} J_{\omega\mathrm{B}} \\
d_{11} &= m_{\mathrm{A}_i} J_{v\mathrm{A}_i,ci}^T J_{v\mathrm{A}_i,ci} \\
d_{12} &= J_{\omega\mathrm{A}_i,ci}^T R_{\mathrm{WA}_i} \mathcal{I}_{\mathrm{A}_i} R_{\mathrm{WA}_i}^T J_{\omega\mathrm{A}_i,ci} \\
d_{13} &= J_{\omega\mathrm{B}}^T R_{\mathrm{WA}_i} \mathcal{I}_{\mathrm{A}_i} R_{\mathrm{WA}_i}^T J_{\omega\mathrm{B}}
\end{aligned} \tag{3.65}$$

It's easy to see that $d_5 = d_9^T$ and $d_8 = d_{10}^T$, therefore, using the convention $d_{1,2} = d_1 + d_2$ the inertia matrix becomes

$$D_{\mathrm{AB}} = \begin{bmatrix} d_{11,12} & d_{5,8} \\ d_{5,8}^T & d_{1,2,3,4,6,7,13} \end{bmatrix} \tag{3.66}$$

where every term d that contains an arm link subscript i is summed for all the joints from $i = 1 : n$. The full Lagrangian becomes

$$\mathcal{L} = \mathcal{K}_{\mathrm{AB}} - \mathcal{P}_{\mathrm{A}} \tag{3.67}$$

and the full dynamical equations is found by substituting (3.67) into (3.17) [64, 65, 81]. And for $\boldsymbol{q} = \begin{bmatrix} \boldsymbol{q}_{\mathrm{A}}^T & \boldsymbol{q}_{\mathrm{B}}^T \end{bmatrix}^T$ the final equations of motion can be expressed as

$$D(\boldsymbol{q})\ddot{\boldsymbol{q}} + C(\boldsymbol{q}, \dot{\boldsymbol{q}}) + G(\boldsymbol{q}) = \boldsymbol{\tau} + A_{manip}(\boldsymbol{q})\lambda \tag{3.68}$$

where $A_{manip}(\boldsymbol{q})\lambda$ are the non-holonomic constraints in (3.17), $C(\boldsymbol{q}, \dot{\boldsymbol{q}})$ is the coriolis and centrifugal terms, and $G(\boldsymbol{q})$ is the gravity vector. Additional external forces can be included such as joint friction or the applied external wrench from (3.10).

3.3 Optimal Control

Optimal control is the description of a control policy that when executed causes the state to follow an optimal process defined by a functional, subjected to given constraints. However, the first challenge is determining if for given a functional, does an optimal process (or trajectory through the state space) exists. To address this, the essentials of optimization are presented followed by applications to cost functional's for control objectives, then finally methods to solve for local or global optimal control policies is presented.

3.3.1 Essentials of Optimization

When solving an optimization problem over a set or given functional, it is assumed that the global optimum (usually in convex optimization) or local optimum (usually in nonlinear optimization) is reachable from the given initial conditions. One way to ensure this is by the careful design of the functional. A set $C \subset \mathbb{R}^n$ is considered affine if a line segment connecting any two points in the set C also lies in the set. Extending this to geometric regions defined by the combination of surfaces (of dimension $\mathbb{R}^{n-1}$ in n-dimensional space), a geometric region is convex if the set of points inside the region and on the defining surfaces are affine.

Given a convex region in $\mathbb{R}^n$, an optimization problem's solution requires finding a point or set of points that maximize or minimize as specified a stated objective given constraints. For a linear programming (LP) problem, consider a vector field $\boldsymbol{v}$ and state space $\boldsymbol{x} \in \mathbb{R}^n$ and a convex polytope defined by

$$A\boldsymbol{x} \leq \boldsymbol{b} \tag{3.69}$$

where the rows of A specify normal vectors for the polytope planes and the inequality with vector

$\boldsymbol{b}$ ensures that the point $\boldsymbol{x}$ is interior to the defining planes. The linear problem is formally stated as

$$\begin{aligned} &\min_{\boldsymbol{x}} \boldsymbol{v}^T\boldsymbol{x} \\ &\textit{subj. to } A\boldsymbol{x} \leq \boldsymbol{b} \\ &\qquad F\boldsymbol{x} = \boldsymbol{h} \end{aligned} \tag{3.70}$$

This states, given a vector field $\boldsymbol{v}$, find the point interior to the convex polytope defined by (3.69) constrained by the equality $F\boldsymbol{x} = \boldsymbol{h}$ that is in the most 'negative' direction of the vector field. When searching for a unique solution, an inherent limitation to LP is that if the vector field $\boldsymbol{v}$ is aligned with a row of A, there may be a set of points $\boldsymbol{x}$ that satisfy the LP and lie on the face of the polytope defined by the corresponding row of A.

A general quadratic programming (QP) problem is formally stated as

$$\begin{aligned} &\min_{\boldsymbol{x}} \boldsymbol{x}^T Q\boldsymbol{x} + \boldsymbol{v}^T\boldsymbol{x} + a \\ &\textit{subj. to } A\boldsymbol{x} \leq \boldsymbol{b} \\ &\qquad F\boldsymbol{x} = \boldsymbol{h} \end{aligned} \tag{3.71}$$

where a is a scalar, and the quadratic term is $\boldsymbol{x}^T Q\boldsymbol{x}$ and describes a paraboloid, and since the objective is to find the minimum of $\boldsymbol{x}$, the matrix Q is defined to be positive semi-definite meaning $|Q| \geq 0$. While this is the most common approach used (and works well in most cases), it lacks a degree of elegance. This is because while the quadratic term minimizes the norm of $\boldsymbol{x}$, the term $\boldsymbol{v}^T\boldsymbol{x}$ minimizes $\boldsymbol{x}$ in the vector field $\boldsymbol{v}$ and these objectives are linearly constrained inside of a polytope defined by the union of half-spaces $A\boldsymbol{x} \leq \boldsymbol{b}$ which can be subject to the same multiple solution scenario as the LP problem (consider a plane that restricts further minimization of the weighted norm $\boldsymbol{x}^T Q\boldsymbol{x}$ and is aligned with $\boldsymbol{v}$). The use of quadratic constraints are captured in the following definition.

The second order cone programming (SOCP) problem is formally stated as

$$\begin{aligned} &\min_{\boldsymbol{x}} \boldsymbol{v}^T \boldsymbol{x} \\ \textit{subj. to } &||A_i \boldsymbol{x} + \boldsymbol{b}_i||_2 \leq \boldsymbol{c}_i^T \boldsymbol{x} + a_i,\ i = 1, \ldots, m \\ &F\boldsymbol{x} = \boldsymbol{h} \end{aligned} \tag{3.72}$$

where a norm cone is defined by the set $C = \{(\boldsymbol{x}, t_{socp})|\ ||x|| \leq t_{socp}\} \subset \mathbb{R}^{n+1}$ and the inequality constraint in (3.72) is a norm cone. Simply stated, the norm cone lies in a space one dimension higher than the state space $\boldsymbol{x}$ and the norm of $\boldsymbol{x}$ is bounded by a scalar (here t_{socp}) on the last dimension. First it must be noted that when $\boldsymbol{c}_i = \boldsymbol{0}$ this reduces to a quadratically constrained quadratic program (QCQP), and if $A_i = 0\ \forall i$ then this reduces to a general LP. Geometrically, the SOCP can be thought of as m intersecting ellipsoids in $\mathbb{R}^n$ whose shapes and projections are determined by A_i and are subjected to linear constraints defined by the combination of $A_i, \boldsymbol{b}_i, \boldsymbol{c}_i$ and norm cone bounded by a combination of $\boldsymbol{b}_i, a_i$. Given the overlapping ellipsoids and hyperplanes, minimization of $\boldsymbol{v}^T\boldsymbol{x}$ within the combined convex region finds the optimum point over the ellipsoids. It is interesting to note that a constant vector field $\boldsymbol{v}$ over a set of overlapping ellipsoids (with bounded norm) will find a unique minimum due to the natural curvature of the ellipsoids [73].

To solve these optimization problems, if it is known that a global unique solution exists and the functions are C^1 differentiable, then it can be solved for directly using Lagrange multipliers. For example, given a function $f(\boldsymbol{x})$ and constraint functions $g_{opt}(\boldsymbol{x}) = 0$, then by defining $h(\boldsymbol{x}, \lambda) = f(\boldsymbol{x}) + \lambda g_{opt}(\boldsymbol{x})$, then differentiating, then the extremal point is satisfied at the unique point $\nabla h(\boldsymbol{x}^*, \lambda^*) = 0$. However, often it is the case that the functions and constraints cannot be easily solved in this manner, and if so it is not computationally feasible. In these scenarios, numerical methods can be used to find the optimal points. Simplistically, this is done using gradient descent methods, where the perturbation $\Delta\boldsymbol{x}$ which decreases $f(\boldsymbol{x})$ the most while respecting constraints is used to update the estimated optimal state $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \gamma\Delta\boldsymbol{x}$. This is done until the value converges for a given step size γ. There are a host of algorithms to perform this, including but not limited to Dantzig's simplex algorithm, Newton's method, the Gauss-Newton algorithm (simplification of Newton's method), and interior point method (an application of Newton's method). Dantzig's simplex algorithm is useful for LP's where the minimized vector field is not aligned with a hyper-plane in (3.70). The solution is assumed to be on one of the finite vertices of the defined, closed polytope. Starting on a vertex, the edges are followed until the optimal vertex is reached, comparing the functional evaluation at each consecutive function to determine next best step. Newton's method is found by taking a second order Taylor series expansion about the current state (single dimension) x_k: $f(x_k + \Delta x) \approx f(x_k) + f^{(1)}(x_k)\Delta x + \frac{1}{2}f^{(2)}(x_k)\Delta x^2$. By differentiating with respect to Δx, and setting this to zero, the step size $\Delta x = -f^{(1)}(x_k)/f^{(2)}(x_k)$ is obtained and used to find the optimum. Newton's method generalized to a vector $\boldsymbol{x} \in \mathbb{R}^n$ becomes

$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n - \gamma \nabla^2 f^{-1}(\boldsymbol{x}_n) \nabla f(\boldsymbol{x}_n) \tag{3.73}$$

where $\gamma \in [0, 1]$ allows for a smaller step-size if desired along the search direction

$$\Delta \boldsymbol{x} = -\nabla^2 f^{-1}(\boldsymbol{x}_n) \nabla f(\boldsymbol{x}_n). \tag{3.74}$$

Newtons method has the requirement that the function be at least twice differentiable. For functions $f_i(x)$ that are only once differentiable the Gauss-Newton method can be used as only the first derivative is required. By defining a least squares problem $f(x) = \frac{1}{2}\sum_i f_i(x)^2$, the Gauss-Newton search direction becomes

$$\Delta \boldsymbol{x} = -\left(\sum_i \nabla f_i(\boldsymbol{x}) \nabla f_i(\boldsymbol{x})^T\right)^{-1} \left(\sum_i f_i(\boldsymbol{x}) \nabla f_i(\boldsymbol{x})\right) \tag{3.75}$$

as developed in [73]. The interior point method is an implementation for convex optimization problems where the objective functions are convex and at least twice differentiable. The value in the interior point method is that it reduces an optimization problem with linear equalities and inequalities to just a problem with linear equalities. This is commonly done using barrier method, where a barrier function (e.g. differentiable logarithmic barrier) is used to represent the inequality as a part of the objective function, where states very close to the barrier are heavily penalized [73].

3.3.2 Essentials of Optimal Control

The goal of optimal control is to find the input policy $\boldsymbol{u}^*(t) \in U$ which causes the system $\dot{\boldsymbol{x}}(t) = f(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{\beta}(t), t)$ to follow a trajectory that minimizes the performance metric $S(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\beta}, t)$. Where u is the input and U is the set of valid control inputs, $\boldsymbol{x}$ is the state of the system, $f(\,\cdot\,)$ describes the state dynamics and β are system parameters that may be either constant or varying, and $c(\,\cdot\,)$ describes state and input constraints. Formally the optimal control problem can be expressed as

$$\begin{aligned} \min_{u,\; t\in[t_0,t_f]} S(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{\beta}(t), t) &= \min_{u,\; t\in[t_0,t_f]} \left(m(\boldsymbol{x}(t_f), \boldsymbol{u}(t_f), t_f) + \int_t^{t_f} g(\boldsymbol{x}(s), \boldsymbol{u}(s), s) ds \right) \\ \textit{subj to}\;\; \dot{\boldsymbol{x}} &= f(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{\beta}(t), t) \\ c(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{\beta}(t), t) &\leq 0 \end{aligned} \tag{3.76}$$

where $m(\,\cdot\,)$ is the terminal cost function and g is the running cost function. The input that satisfies (3.76) is the optimal policy $\boldsymbol{u}^*(t)$ over the entire interval $t \in [t_0, t_f]$. This leads to the Bellman Principle of Optimality [9] which states that “if $\boldsymbol{u}^*(t)$ is an optimal policy over $t \in [t_0, t_f]$ with an initial state $\boldsymbol{x}(t_0)$, then $\boldsymbol{u}^*(t)$ is necessarily optimal over the subinterval $[t_0 + \Delta t, t_f]$ for any Δt such that $t_f - t_0 \geq \Delta t > 0$." The task becomes finding the control input that satisfies the principle, to do this, consider the minimum cost function

$$S^*(\boldsymbol{x}, t) = \min_{u,\; t\in[t_0,t_f]} \left(m(\boldsymbol{x}(t_f), \boldsymbol{u}(t_f), t_f) + \int_t^{t_f} g(\boldsymbol{x}(s), \boldsymbol{u}(s), s) ds \right) \tag{3.77}$$

which is satisfied by the optimal input $u^*(t)$, by subdividing the integration integral and using the principle of optimality

$$S^*(\boldsymbol{x}(t), t) = \min_{u,\; t\in[t_0,t_f]} \left(\int_t^{t+\Delta t} g(\boldsymbol{x}(s), \boldsymbol{u}(s), s) ds + S^*(\boldsymbol{x}(t+\Delta t), t+\Delta t) \right) \tag{3.78}$$

Assuming $S^*(\boldsymbol{x}(t), t)$ is differentiable (partial derivatives exist and are bounded) then $S^*(\boldsymbol{x}(t + \Delta t), t + \Delta t)$ then a Taylor series can be expanded about the point $S^*(\boldsymbol{x}(t), t)$

$$\begin{aligned} S^*(\boldsymbol{x}(t), t) = \min_{u,\ t\in[t_0,t_f]} \Bigg(& \int_t^{t+\Delta t} g(\boldsymbol{x}(s), \boldsymbol{u}(s), s)ds + S^*(\boldsymbol{x}(t), t) + \frac{\partial S^*}{\partial t}(\boldsymbol{x}(t), t)\Delta t \\ & + \left(\frac{\partial S^*}{\partial \boldsymbol{x}}(\boldsymbol{x}(t), t)\right)^T [\boldsymbol{x}(t + \Delta t) - \boldsymbol{x}(t)] + \mathcal{O}\Bigg) \end{aligned} \tag{3.79}$$

where $\mathcal{O}$ are higher order terms. Next, a very small Δt is considered and used to find the optimal input, which is the powerful assertion behind the principle of optimality because the problem of finding the optimal control over the whole interval has been reduced to finding the optimal control over the reduced interval $[t, t + \Delta t]$. With the small Δt (3.79) can be simplified to

$$\begin{aligned} S^*(\boldsymbol{x}(t), t) = \min_{u} \Bigg(& g(\boldsymbol{x}(t), \boldsymbol{u}(t), t)\Delta t + S^*(\boldsymbol{x}(t), t) + \frac{\partial S^*}{\partial t}(\boldsymbol{x}(t), t)\Delta t \\ & + \left(\frac{\partial S^*}{\partial \boldsymbol{x}}(\boldsymbol{x}(t), t)\right)^T \dot{\boldsymbol{x}}(t) + o(\Delta t)\Bigg) \end{aligned} \tag{3.80}$$

where the derivative $\dot{\boldsymbol{x}}$ is defined in (3.76) and $o(\Delta t)$ are the higher order terms. After canceling the $S^*(\boldsymbol{x}(t), t)$ and dividing all by Δt, the higher order terms vanish by the principle $\lim_{\Delta t \to 0} \left|\frac{o(\Delta t)}{\Delta t}\right| = 0$. Leaving the following conditions

$$\begin{aligned} 0 = \frac{\partial S^*}{\partial t}(\boldsymbol{x}(t), t) + \min_{u} \Bigg(& g(\boldsymbol{x}(t), \boldsymbol{u}(t), t) \\ & + \left(\frac{\partial S^*}{\partial \boldsymbol{x}}(\boldsymbol{x}(t), t)\right)^T [f(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{\beta}(t), t)]\Bigg) \end{aligned} \tag{3.81}$$

$$S^*(\boldsymbol{x}(t_f), t_f) = m(\boldsymbol{x}(t_f), \boldsymbol{u}(t_f), t_f) \tag{3.82}$$

The argument that requires the minimization of the control input is called the Hamiltonian

$$\begin{aligned}\mathcal{H}(\boldsymbol{x}(t),\boldsymbol{u}(t),\nabla_x S^*,t) &\triangleq g(\boldsymbol{x}(t),\boldsymbol{u}(t),t)\\ &\quad+\left(\frac{\partial S^*}{\partial \boldsymbol{x}}(\boldsymbol{x}(t),t)\right)^T[f(\boldsymbol{x}(t),\boldsymbol{u}(t),\boldsymbol{\beta}(t),t)] \qquad (3.83)\\ \mathcal{H}(\boldsymbol{x}(t),\boldsymbol{u}^*(\boldsymbol{x}(t),\nabla_x S^*,t),\nabla_x S^*,t) &\triangleq \min_u \Big(g(\boldsymbol{x}(t),\boldsymbol{u}(t),t)\\ &\quad+\left(\frac{\partial S^*}{\partial \boldsymbol{x}}(\boldsymbol{x}(t),t)\right)^T[f(\boldsymbol{x}(t),\boldsymbol{u}(t),\boldsymbol{\beta}(t),t)]\Big) \qquad (3.84)\end{aligned}$$

using (3.84) the Hamilton-Jacobi-Bellman equation is defined as

$$0=\frac{\partial S^*}{\partial t}(\boldsymbol{x}(t),t)+\mathcal{H}\left(\boldsymbol{x}(t),\boldsymbol{u}^*(\boldsymbol{x}(t),\frac{\partial S^*}{\partial \boldsymbol{x}}(\boldsymbol{x}(t),t),t),t\right) \qquad (3.85)$$

It must be noted that while solving (3.85) is a necessary condition it is not always sufficient (an optimal control law will satisfy (3.85), however a control law that simply satisfies (3.85) is not guaranteed to be truly optimal). When $\boldsymbol{u}$ is unbounded and the Hessian of the Hamiltonian is positive definite it is a necessary condition that the optimal control input satisfy

$$\frac{\partial \mathcal{H}}{\partial \boldsymbol{u}}=\mathbf{0} \qquad (3.86)$$

and will minimize $\mathcal{H}$ [32, 66]. For linear systems described by $\dot{\boldsymbol{x}}(t)=A_{lin}\boldsymbol{x}(t)+B_{lin}\boldsymbol{u}(t)$, and performance metric $g(\boldsymbol{x}(t),\boldsymbol{u}(t))=\boldsymbol{x}^TQ\boldsymbol{x}+\boldsymbol{u}^TR_u\boldsymbol{u}$ and boundary condition $m(\boldsymbol{x}(t),\boldsymbol{u}(t))=\boldsymbol{x}^TM\boldsymbol{x}$, where M, Q are real, positive semi-definite and R_u is positive definite then the control input that satisfies (3.86) and the boundary condition (3.82) is

$$\boldsymbol{u}^*(t)=-R_u^{-1}B_{lin}^T\frac{\partial S^*}{\partial \boldsymbol{x}}(\boldsymbol{x}(t),t) \qquad (3.87)$$

the task becomes finding the suitable solution for $\frac{\partial S^*}{\partial \boldsymbol{x}}(\boldsymbol{x}(t),t)$, informed by the fact that a quadratic function of the state produces the minimum cost for a discrete linear regulator problem a valid solution choice is

$$\frac{\partial S^*}{\partial \boldsymbol{x}}(\boldsymbol{x}(t),t)=\frac{1}{2}\boldsymbol{x}^T(t)K_{ric}(t)\boldsymbol{x} \qquad (3.88)$$

where $K_{ric}(t)$ is a real, symmetric positive-definite matrix. Substituting (3.88) into (3.87) and that into (3.81) with boundary condition (3.82), and since these must hold for all $\boldsymbol{x}$ after matrix manipulation obtain the matrix Riccati equation and final boundary value for $K_{ric}(t)$

$$-\dot{K}_{ric}(t) = Q(t) - K_{ric}(t)B_{lin}(t)R_u^{-1}(t)B_{lin}^T(t)K_{ric}(t) + K_{ric}(t)A_{lin}(t) + A_{lin}^T(t)K_{ric}(t) \tag{3.89}$$

$$K_{ric}(t_f) = M \tag{3.90}$$

solving the matrix Riccati equation is done by solving the system of linear equations

$$\begin{bmatrix} \dot{X} \\ \dot{Y} \end{bmatrix} = \begin{bmatrix} A_{lin} & -B_{lin}R_u^{-1}B_{lin}^T \\ -Q & -A_{lin}^T \end{bmatrix} \begin{bmatrix} X \\ Y \end{bmatrix} \tag{3.91}$$

where

$$K_{ric}(t) = Y(t)X^{-1}(t) \tag{3.92}$$

Using the chain rule and substitution its straight-forward to show that (3.92) with (3.91) satisfy (3.89). When the matrices A_{lin}, B_{lin}, Q, R_u are time varying then (3.91) is time varying and an analytical solution is not straight-forward, however if they are time-invariant then solutions can be found in terms of matrix exponential functions [18]. And the optimal control law becomes

$$\boldsymbol{u}^*(t) = -R_u^{-1}(t)B_{lin}^T(t)K_{ric}(t)\boldsymbol{x}(t) \tag{3.93}$$

Usually to ensure a unique solution exists and use common gradient descent methods $g(\boldsymbol{x}(t), \boldsymbol{u}(t), t)$ is chosen to be differentiable and represent a convex (or is approximated as convex for local optimality). This usually implies that the goal is therefore to drive $\boldsymbol{x}(t)$ to a nominal value, which can be done with a simple change of coordinates to make a set-point $\boldsymbol{x}_0$ the origin. If however the goal is to track a desired trajectory, then the set-point is no longer static, and the control objective can be stated in terms of the error

$$\boldsymbol{e}_x(t) = \boldsymbol{x}_d(t) - \boldsymbol{x}(t), \tag{3.94}$$

then the goal is to drive the error to zero, and the control objective $\boldsymbol{u}^*(t)$ becomes an explicit function of (3.94). It must be noted in this case that the control input is only activated when the error is non-zero, meaning that the performance metric will always be greater than the optimum unless the desired trajectory can be estimated, and can be taken into account in a feed-forward term. Another consideration is when there is uncertainty in the state

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{\beta}(t), t) + L_\eta(\boldsymbol{x}(t), t)\boldsymbol{\eta}(t) \tag{3.95}$$

where $\boldsymbol{\eta}(t)$ is additive noise whose actual value is unknown and $L_\eta(\boldsymbol{x}(t), t)$ defines its projection into the current state space. When (3.95) is substituted into (3.76) as the dynamical constraint this forms the problem statement for Stochastic control. In this case, if the noise disturbances $\boldsymbol{\eta}(t)$ are unknown, then the deterministic performance metric cannot be minimized by the choice of control. However if the statistics of $\boldsymbol{\eta}(t)$ are known (e.g. has a zero mean and Gaussian distribution), then a performance metric based on the expected (mean) value of the previous performance metric can be minimized [66]. Both of these problems require prediction of a future state in order to be optimal, which motivates predictive control as an online optimization problem which solves for the optimal control informed by the models of the system dynamics $f(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{\beta}(t), t)$, expected desired trajectory $\boldsymbol{x}_d(t)$, and disturbances $\boldsymbol{\eta}(t)$ [66, 75].

3.4 Model Predictive Control

The general description of model predictive control (MPC) is the design of a controller that minimizes a performance metric (3.76) using either *a priori* or online estimated knowledge of the dynamical model, the associated uncertainty in the state, and prescribed state trajectories. With such a general description of MPC, there are many algorithms that it encapsulates including but not limited to dynamic matrix control, model algorithmic control, inferential control, and internal model control [21, 44].

The power behind MPC is that it relies heavily on the principle of optimality, in that if the optimal control can be continually calculated over a finite time horizon, then the total control over the entire trajectory will be the optimal control. When the desired state trajectory is described by a set-point (constant), and the true dynamics of the system are known and there is no uncertainty or disturbance in the state, then the MPC problem reduces to solving the Hamilton-Jacobi-Bellman (principle of optimality) equation (3.85) [44]. One way to handle feasibility constraints is to use barrier functions in the objective which also improves computation when optimizing over the control input [75]. When there is noise or disturbance $\boldsymbol{\eta}(t)$ added to the system as in (3.95) whose statics can be modeled, the principle of optimality is updated to include that uncertainty as

$$\begin{aligned} 0 = &\frac{\partial S^*}{\partial t}(\boldsymbol{x}(t), t) + \mathcal{H}(\boldsymbol{x}(t), \boldsymbol{u}^*(\boldsymbol{x}(t), \frac{\partial S^*}{\partial \boldsymbol{x}}(\boldsymbol{x}(t), t), t), \nabla_x S^*, t) \\ &+ \frac{1}{2} Tr\left(\frac{\partial^2 S^*}{\partial \boldsymbol{x} \partial \boldsymbol{x}} L_\eta(\boldsymbol{x}(t), t) W_\eta(t) L_\eta^T(\boldsymbol{x}(t), t)\right) \end{aligned} \tag{3.96}$$

where $Tr(\,\cdot\,)$ is the matrix trace operator, $L_\eta(\boldsymbol{x}(t), t)$ is as defined in (3.95), and

$$\mathbb{E}\left[\boldsymbol{\eta}(t)\right] = \bar{\boldsymbol{\eta}} = 0 \tag{3.97}$$

$$\mathbb{E}\left[\boldsymbol{\eta}^T(t)\boldsymbol{\eta}(t)\right] = \boldsymbol{\eta}(t) \tag{3.98}$$

where $\mathbb{E}[\,\cdot\,]$ is the expectation operator, and (3.97) is the zero mean for Gaussian white noise and (3.98) is the covariance [66]. Note that the argument of the trace in last term is positive semi-definite, with zero being achieved only when there is no uncertainty over the disturbance or noise. Since by definition $\boldsymbol{\eta}(t)$ is not explicitly known then this greater than the deterministic principle of optimality over the trajectory as expected.

The problems of estimating the system dynamical model and estimating the desired trajectory are essentially equivalent as a change in state using (3.94) would encapsulate both problems with the goal of driving the state to the origin. The new state will be referred to as the error, but embodies both the error in dynamical model parameter estimation as well as desired trajectory. If the time variant error function $\boldsymbol{e}(t)$ of the system is unknown and is impossible or difficult to estimate, then for relatively small perturbations robust control provides bounded performance. Traditionally, robust control is the study of characterizing and bounding the systems worst performance over a specified interval (formally using stability criterion), with controller design based on a pre-specified performance metric termed as optimal synthesis [10]. To do this, the resulting error or its statistics must be characterized or no such bounds on performance can be found. When only the statistics are available, if a robust contraction constraint is enforced (causing worse case prediction of the state or performance cost to contract over the interval), then nominal performance may be achieved [44].

It is often challenging to enforce such constraints, and for large and growing errors it is often best to actively estimate the underlying models that describe the system dynamics or the desired trajectory. So in the cases where the states and effective model parameters are observable, improved approximation of these models for the dynamics and desired trajectories will lead to better overall performance and approach optimal control. To approximate a model, valid estimates of the state or parameters as well as their processing must be obtained to update the model. The next session presents methods and theory behind state estimation.

3.5 State Estimation

In order to attenuate the state of a generic system (denoted as $\boldsymbol{x}(t)$), the state it must be at least partially and preferably fully observable. These observations in robotics are usually obtained using sensors that have a degree of uncertainty in measurements. This makes it imperative to be able to model the statistics of these uncertainties in order better model the system and ultimately control the system. The multivariate probability density function will be denoted as $pr(\,\cdot\,)$ which must satisfy

$$\int pr(\boldsymbol{x})d\boldsymbol{x} = 1 \tag{3.99}$$

$pr(\boldsymbol{x})$ expresses the likelihood that $\boldsymbol{x}$ will fall in the volume $d\boldsymbol{x}$, hence integrating over the entire space the probability is 1. The expectation operator is denoted as $\mathbb{E}[\,\cdot\,]$, and for a function $f(\boldsymbol{x})$ under a probability distribution $pr(\boldsymbol{x})$ is defined as

$$\mathbb{E}[f] = \int pr(\boldsymbol{x})f(\boldsymbol{x})d\boldsymbol{x} \tag{3.100}$$

where $f(\boldsymbol{x}) = \boldsymbol{x}$ is valid. The covariance is defined as

$$cov[\boldsymbol{x}, \boldsymbol{y}] = \mathbb{E}_{x,y}\left[(\boldsymbol{x} - \mathbb{E}[\boldsymbol{x}])\left(\boldsymbol{y}^T - \mathbb{E}[\boldsymbol{y}^T]\right)\right] \tag{3.101}$$

where $\boldsymbol{x}, \boldsymbol{y}$ are vectors and $\boldsymbol{x} = \boldsymbol{y}$ is valid [13]. A covariance of an element $\boldsymbol{x}_i$ or function $f(\boldsymbol{x})$ with itself is called the variance. A Gaussian distribution of $\boldsymbol{x}$ denoted as $\mathcal{N}(\boldsymbol{x}|\bar{\boldsymbol{x}}, \Sigma)$ where $\bar{\boldsymbol{x}}, \Sigma$ are the mean and covariance matrix respectively, is defined as

$$\mathcal{N}(\boldsymbol{x}|\bar{\boldsymbol{x}}, \Sigma) = \frac{1}{\sqrt{2\pi|\Sigma|}} \exp\left(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{u})^T \Sigma^{-1} (\boldsymbol{x} - \boldsymbol{u})\right) \tag{3.102}$$

The Gaussian distribution has the following properties $\int_{-\infty}^{\infty} \mathcal{N}(\boldsymbol{x}|\bar{\boldsymbol{x}}, \Sigma) d\boldsymbol{x} = 1$ and $\mathcal{N}(\boldsymbol{x}|\bar{\boldsymbol{x}}, \Sigma) > 0$. Unless explicitly stated or observed, it is generally assumed that uncertainty in most measurements of physically observed processes exhibit Gaussian noise about a mean. This is due to the Central Limit Theorem which states that for any distribution, as the number of samples approaches infinity the distribution converges to a normal distribution [13].

If it is assumed that the dynamical function or trajectory model has a unique output given particular initial conditions, then when the initial conditions are known, previous states are not required to predict the processing of the state. This lays the foundation for state estimation using Markov processes, as a Markov process predicts a future state independent of past states given the current state of the system. A Hidden Markov Model (HMM) is a process where the actual state $\boldsymbol{x}$ is not observed and is referred to as the latent variable, however there is a projection to observed variables $\boldsymbol{z}$. The Kalman filter is one implementation of a HMM and a class of Bayesian filter, with a latent variable and assumes that all variables (observed and latent) have Gaussian distributions. There are a few extensions of the Kalman filter, two common ones are the Extended Kalman filter (EKF) and the Unscented Kalman filter (UKF), both of these improve on the approximation of the processing dynamics as shown in (3.95). These filters make the assumption that the state is measured discretely (which is the case using hardware), however the Kalman-Bucy filter provides a continuous time representation of the Kalman Filter. Due to its generality, the UKF is presented first, to handle non-linearity in the state-space the covariance is used to sample the local space based on the uncertainty using the following function

$$X = \sigma(\hat{\boldsymbol{x}}, \Sigma) = \begin{bmatrix} \hat{\boldsymbol{x}} & \hat{\boldsymbol{x}} + \gamma\sqrt{\Sigma} & \hat{\boldsymbol{x}} - \gamma\sqrt{\Sigma} \end{bmatrix} \tag{3.103}$$

The estimated (or latent) variable in (3.103) is $\hat{\boldsymbol{x}}$ with associated uncertainty in the covariance matrix Σ and the columns of the resulting matrix are referred to as the sigma points. For a n-dimensional state space there are $2n+1$ sigma points in (3.103) defined by the columns of $\sqrt{\Sigma}$ with γ as a constant variable defined as $\gamma = \alpha^2(n+\kappa)$ where α, κ are adjustable, scalar parameters. The Kalman filter is described by an prediction step and an update step, for the UKF these are the prediction step:

$$\begin{aligned}
\chi &= f(\chi_{t-1}, \boldsymbol{u}_{t-1}) \\
\hat{\boldsymbol{x}}_t &= \mathbb{E}\left[\chi_t\right] \\
\Sigma_t &= cov\left[\left(\chi_t - \hat{x}\right), \left(\chi_t - \hat{x}\right)\right] + R_{kf,t} \\
\chi_t &= \sigma(\hat{\boldsymbol{x}}_t, \Sigma_t)
\end{aligned} \tag{3.104}$$

and for the update step:

$$\begin{aligned}
Z_{kf} &= h(\chi_t) \\
\hat{z}_t &= \mathbb{E}\left[Z_{kf}\right] \\
L_{kf,t} &= cov\left[\left(Z_{kf} - \hat{z}\right), \left(Z_{kf} - \hat{z}\right)\right] + Q_{kf,t} \\
M_{kf,t} &= cov\left[\left(X_t - \hat{x}_t\right), \left(Z_{kf} - \hat{z}\right)\right] \\
K_{kf,t} &= M_{kf,t} L_{kf,t}^{-1} \\
\hat{x}_t &= \hat{x}_t + K_{kf,t}(z_t - \hat{z}_t) \\
\Sigma_t &= \Sigma_t - K_{kf,t} L_{kf,t} K_{kf,t}^T
\end{aligned} \tag{3.105}$$

where $\boldsymbol{u}$ is the control input, $\boldsymbol{z}$ is the measurement which is observed by the sensors, and $R_{kf,t}$, $Q_{kf,t}$ are the uncertainties associated with the process and the measurement respectively [70]. The traditional Kalman-Bucy filter was designed for linear time invariant systems, but many renditions exist including the hybrid Kalman-Bucy filter that has both continuous and discrete components, Extended Kalman-Bucy filter, and finally the continuous time Unscented Kalman-Bucy filter (UKBF) presented here for a system described by the dynamics and output

$$\begin{aligned}\dot{\boldsymbol{x}}(t) &= f(\boldsymbol{x}(t), \boldsymbol{u}(t), t) + M_{kf}(t)\boldsymbol{\eta}_{R_{kf}} \\ \boldsymbol{z}(t) &= h(\boldsymbol{x}(t), t) + L_{kf}(t)\boldsymbol{\eta}_{Q_{kf}}\end{aligned} \tag{3.106}$$

where $\boldsymbol{\eta}_{Q_{kf}}$,$\boldsymbol{\eta}_{R_{kf}}$ are white noises with spectral densities $\boldsymbol{\eta}_{Q_{kf}} \sim \mathcal{N}(\mathbf{0}, Q_{kf})$ and $\boldsymbol{\eta}_{R_{kf}} \sim \mathcal{N}(\mathbf{0}, R_{kf})$ respectively and projection matrices $L_{kf}(t)$, $M_{kf}(t)$. Then with a weighting matrix as a function of the weightings γ: $W_\sigma = \Gamma_{kf}(\gamma)$ for the sigma points as defined in [59], the complete continuous time UKBF can be represented as

$$\begin{aligned}\frac{d\hat{\boldsymbol{x}}}{dt} &= f(X(t), \boldsymbol{u}(t), t) + K_{kf}\left(Z_{kf} - h(\chi(t), t)\right) \\ \frac{d\Sigma}{dt} &= X(t)W_\sigma f^T\left(\chi(t), \boldsymbol{u}(t), t\right) + f\left(\chi(t), \boldsymbol{u}(t), t\right)W_\sigma X(t) \\ &\quad + L_{kf}Q_{kf}L_{kf}^T - K_{kf}M_{kf}R_{kf}M_{kf}^T K_{kf}^T \\ K_{kf} &= X(t)W_\sigma h^T(\chi(t), t)Q_{kf}^{-1}\end{aligned} \tag{3.107}$$

as shown in [59, 86]. So these methods present a way to model the procession of states and the associated uncertainty given a valid dynamical model. However, if the estimating model is poor then the associated error can be expected to reflect the quality of modeling, in fact for drastically incorrect models, the white noise assertion in (3.106) becomes incorrect. Hence the challenge becomes improving the model representation during operation which will in turn improve estimation which will then in turn improve control. This motivates the need for model approximation, and that performed online in the form of adaptive control through model approximation.

3.6 Model Approximation

In order to estimate a processing state accurately, it is important to have a valid model as this will directly effect the accumulated errors and therefore the ability to optimally control the system. Let

Table 3.1: Modeling types and estimation techniques [47].

Model type	Estimation Architecture
Forward Model	Direct modeling
Inverse Model	Direct modeling, indirect modeling
Mixed Model	Direct modeling*, indirect modeling, distal teacher

a system process and output model be described by

$$\begin{aligned} \dot{\boldsymbol{x}}(t) &= f(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{\beta}(t), t) \\ \boldsymbol{z}(t) &= h\boldsymbol{x}(t), \boldsymbol{\beta}(t), t \end{aligned} \tag{3.108}$$

where $\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{\beta}$ are the state, output and time varying model parameters (present in the parametric model). There are three major types of modeling techniques: forward modeling, inverse modeling and mixed modeling. Forward modeling is defined as using previous states and current inputs via a model to predict future states. Inverse modeling is defined as using previous states and future desired or expected states via a model to determine required inputs. Mixed modeling is a combination of forward and inverse modeling by using both previous state and previous inputs as well as future desired or expected states to determine the current and next inputs.

Common methods of estimating these models include direct modeling, indirect modeling and the distal teacher method. In direct modeling, a model is estimated by observing the inputs and resulting outputs, essentially mapping actions to change in the state and adjusting the model based on approximation errors (predictions and resulting outputs). In indirect modeling, a feedback error measure is defined, and the model is estimated by minimizing this error metric. In distal teacher modeling, the forward model is estimated from observing inputs and resulting outputs, and this is used to update (or teach) the inverse model by minimizing the error between the expected output from the forward model and the resulting output from the current inverse model, with the inverse model being the true inverse of the forward model if this error is minimized. The inverse model is then used to perform control on the actual system. The different model types and their corresponding estimation architectures are shown in Table 3.1, where for the mixed model the direct modeling architecture is appropriate if it is known that the proposed model is invertible [47].

When the model is parametric as in (3.108), in each of the above estimation architectures, an error metric is to be minimized by adjusting the available parameters $\boldsymbol{\beta}(t)$ which could be constant or varying depending on the system. In order to see the effect of changing $\boldsymbol{\beta}(t)$, the fundamental qualities of $f(\,\cdot\,)$ in (3.108) must be understood. It also important to note, that parametric models have the inherent limitation that they are subject to the structure of $f(\,\cdot\,)$ regardless of the parameters $\boldsymbol{\beta}(t)$ chosen, and if this modeling is fundamentally wrong then the system approximation can be expected to have persistent errors, which is the motivation for non-parametric modeling. First the residual must be defined, the approximation error $\boldsymbol{e}_x(t)$ defined as the difference between the real state $\boldsymbol{x}(t)$ and the estimated state $\hat{\boldsymbol{x}}(t)$

$$\begin{aligned} \boldsymbol{e}_x(t) &= \boldsymbol{x}(t) - \hat{\boldsymbol{x}}(t) && (3.109)\\ &= \boldsymbol{x}(t) - \int_{t_0}^{t_0+\Delta t} f(\boldsymbol{x}(s), \boldsymbol{u}(s), \boldsymbol{\beta}(s), s)ds && (3.110)\\ &= \boldsymbol{x}(t) - \int_{0}^{\Delta t} f(\boldsymbol{x}(s), \boldsymbol{u}(s), \boldsymbol{\beta}(s), s)ds - \boldsymbol{x}(t_0) && (3.111) \end{aligned}$$

Then for very small Δt, this can be approximated as

$$\boldsymbol{e}_x(t) = \dot{\boldsymbol{x}}(t)\Delta t - f(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{\beta}(t), t)\Delta t + \boldsymbol{e}_x(t - \Delta t) \tag{3.112}$$

$$\boldsymbol{e}_x(t) = \left(\dot{\boldsymbol{x}}(t) - f(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{\beta}(t), t)\right)\Delta t + \boldsymbol{e}_x(t - \Delta t) \tag{3.113}$$

Note that if the residual $(\dot{\boldsymbol{x}}(t) - f(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{\beta}(t), t))$ is minimized or decreased over the whole time interval $\forall t$, then the approximation error will also be minimized or decreased over the entire time interval. Note that the same development can be performed for the output function $(\dot{\boldsymbol{z}}(t) - h\boldsymbol{x}(t), \boldsymbol{\beta}(t), t))$. If $f(\,\cdot\,), h\,\cdot\,)$ are smooth and differentiable with respect to the parameters $\boldsymbol{\beta}(t)$, then with an squared residual error metric defined as

$$\begin{aligned} e_{res}(\boldsymbol{\beta}, t) &= \left(\dot{\boldsymbol{x}}(t) - f(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{\beta}(t), t)\right)^2 \\ e_{res}(\boldsymbol{\beta} + \Delta\boldsymbol{\beta}, t) &= e_{res}(\boldsymbol{\beta}, t) + e_{res}(\boldsymbol{\beta}, t)^{(1)}\Delta\boldsymbol{\beta}(t) + \frac{1}{2}e_{res}(\boldsymbol{\beta}, t)^{(2)}\Delta\boldsymbol{\beta}^2(t) && (3.114) \end{aligned}$$

when expanded by the Taylor series, then using Newtons method the change in the parameters $\Delta\boldsymbol{\beta}$ is

$$\Delta\boldsymbol{\beta}(t) = \gamma\left(\dot{\boldsymbol{x}}(t) - f(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{\beta}(t), t)\right)\left(\frac{\partial f(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{\beta}(t), t)}{\partial\boldsymbol{\beta}}\right)^{-1} \tag{3.115}$$

In non-parametric modeling, a statistical association between inputs and outputs are modeled. There are many ways in which this non-parametric modeling can be performed, including but not limited to Neural Nets and Gaussian Processes, but for brevity only Gaussian Processes (GP) will be presented in this section. Given a set of inputs $\boldsymbol{x}_k$ (which could include states, inputs etc) and associated observations $\boldsymbol{y}_k$ (which could be outputs, derivative states etc), a training set can be obtained from the correlated inputs and observations $\boldsymbol{y}_{train}$, $\boldsymbol{x}_{train}$. Defining a generic Gaussian kernel as

$$\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sigma_\kappa^2 \exp\left(\frac{(\boldsymbol{x}_i - \boldsymbol{x}_j)^T(\boldsymbol{x}_i - \boldsymbol{x}_j)}{2l_\kappa^2}\right) \tag{3.116}$$

where σ_κ and l_κ are the variance and length scale respectively. Then a variance matrix K_{train} can be constructed, where the elements of the matrix $K_{ij,train}$ are found using the kernel function in (3.116). For a set of test inputs $\boldsymbol{x}_{test}$, the associated $\boldsymbol{y}_{test}$ is the output of the non-parametric model. All of the statistical modeling methods make the inherent assumption that states that are 'close' defined by a metric distance projected through the kernel function have the same output properties (with the obvious quality that a training point should return the trained output). Therefore the 'closeness' of a test point with the training points can be denoted by the covariance matrix $K_{*,mn}$, for m test points and n training points defined by

$$K_{*,ij} = \kappa(\boldsymbol{x}_{test,i}, \boldsymbol{x}_{train,j}) \tag{3.117}$$

Finally, the covariance matrix for the test points with test points is denoted as $K_{**,mm}$ and is defined as

$$K_{**,ij} = \kappa(\boldsymbol{x}_{test,i}, \boldsymbol{x}_{test,j}) \tag{3.118}$$

Defining a small value δ_n, and mean value function $m(\,\cdot\,)$ the Gaussian Process algorithm can be expressed as

$$\begin{aligned} \boldsymbol{y}_{test} &= m(x_{test}) + K_*^T \left(K_{train} + \delta_n^2 I\right)^{-1} (\boldsymbol{y}_{train} - m(x_{train})) \\ V[\boldsymbol{y}_{test}] &= K_{**} - K_*^T \left(K_{train} + \delta_n^2 I\right)^{-1} K_* \end{aligned} \tag{3.119}$$

Note that the model of the system is based on the training data output $\boldsymbol{y}_{train}$ and the associated covariance matrix of the training data K_{train}. The inherent limitations of such methods are the data smoothness, high dimensionality, noisy data, redundant data, missing data and outliers to name a few [47]. Some methods that are employed to improve on these limitations are using local GP's for high-dimensional data [48], using a mixture of experts for noisy and missing data regions [5], and sparsification for redundant data [46]. However despite all of these methods to improve the non-parametric methods, there is the persistent, inherent limitation that the modeling can only be expected to perform well near (metric closeness or interpolated) training data.

To handle this last limitation, semi-parametric modeling has been proposed as a method to bridge the gaps in both methodologies. In semi-parametric modeling the parametric model can be used to induce a prior (which is helpful when examining test inputs far from the training data) as well as being useful for defining an informed (e.g. physics based) kernel [19]. In this way the strengths of both methods can be leveraged, where the parametric model improves estimation far from the training data and is allowed to improve the parameters, and the non-parametric model allows the system to adapt to observations that are not reflected in the given structure of the parametric model, which over time with more observations improves the system's performance. An example of such a semi-parametric modeling scheme is presented using the GP for the non-parametric modeling component

$$\begin{aligned} \boldsymbol{y}_{test}(t) &= f(\boldsymbol{x}_{test}(t), t) + K_*^T \left(K_{train} + \delta_n^2 I\right)^{-1} (\boldsymbol{y}_{train} - f(\boldsymbol{x}_{train}(t), t)) \\ V[\boldsymbol{y}_{test}] &= K_{**} - K_*^T \left(K_{train} + \delta_n^2 I\right)^{-1} K_* \\ \Delta\boldsymbol{\beta}(t) &= \gamma \left(\frac{\partial f(\boldsymbol{x}_{test}(t), t)}{\partial \boldsymbol{\beta}}\right)^{\dagger} (\boldsymbol{y}_{test}(t) - f(\boldsymbol{x}_{test}(t), t)) \end{aligned} \tag{3.120}$$

Adaptive control is the process of doing the above mentioned model parameter adjustments concurrently during optimal control [60, 64].

3.7 Essentials of Controller Stability Analysis

Given a dynamical system described by (3.108), for this section let $\boldsymbol{x}$ be redefined through a change of coordinates such that the origin $\boldsymbol{x} = \mathbf{0}$ is the control objective. The origin is considered a stable point, if for small scalars $\epsilon, \delta > 0$, for a state starting within $||\boldsymbol{x}(t_0)|| < \delta$, then $||\boldsymbol{x}(t)|| < \epsilon$ true for all $t \geq 0$. The origin is considered asymptotically stable if for $c > 0$, for all $||\boldsymbol{x}(t_0)|| < c$ that $\lim_{t\to\infty} \boldsymbol{x}(t) = 0$. A region of attraction D_A for the origin is defined as points $\boldsymbol{x} \in \mathbb{R}^n$ such that $\lim_{t\to+\infty} \boldsymbol{x}(t) = \mathbf{0}$. Given a dynamical function, the task becomes determining if the origin is stable, this can sometimes be done using the Lyapunov stability theorem. A fundamental part of the Lyapunov stability theorem is the use of the Lie derivative. The Lie derivative of a function along a vector field measures changes in the function along differential equation solutions of the vector field. It is defined here as

$$\mathcal{L}_{lie,f}\mathcal{V}(\boldsymbol{x}) = \frac{\partial \mathcal{V}}{\partial \boldsymbol{x}} \cdot f(\boldsymbol{x}) \tag{3.121}$$

where $f(\boldsymbol{x})$ describes the evolution of the state $\boldsymbol{x}$, and is hence a vector field. If $\boldsymbol{x}(t)$ is a solution to $\dot{\boldsymbol{x}}(t) = f(\boldsymbol{x}(t), t)$ then

$$\mathcal{L}_{lie,f}\mathcal{V}(\boldsymbol{x}(t)) = \dot{\mathcal{V}}(\boldsymbol{x}(t)). \tag{3.122}$$

The Lyapunov time derivative is

$$\begin{aligned}\dot{\mathcal{V}}(\boldsymbol{x}(t)) &= \frac{\partial \mathcal{V}}{\partial t} + \sum_i^n \frac{\partial \mathcal{V}(\boldsymbol{x})}{\partial \boldsymbol{x}_i}\frac{d\boldsymbol{x}_i}{dt} \\ &= \frac{\partial \mathcal{V}}{\partial t} + \sum_i^n \frac{\partial \mathcal{V}(\boldsymbol{x})}{\partial \boldsymbol{x}_i} f_i(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{\beta}(t), t) \end{aligned} \tag{3.123}$$

The Lyapunov stability theorem states that given a function $\mathcal{V} : D \subset \mathbb{R}^n \to \mathbb{R}$ that is continuously differentiable and has the property that $\mathcal{V}(\mathbf{0}) = 0$, $\mathcal{V}(\boldsymbol{x}) > 0 \; \forall \boldsymbol{x} \in D\backslash\{\mathbf{0}\}$ and $\dot{\mathcal{V}} \leq 0 \; \forall \boldsymbol{x} \in D$ then the origin $\boldsymbol{x} = \mathbf{0}$ is stable in D, and if $\dot{\mathcal{V}} < 0 \; \forall \boldsymbol{x} \in D$ then the origin is asymptotically stable in D. However, in some cases for a given $\mathcal{V}$, the property $\dot{\mathcal{V}} < 0$ may not hold over the entire region of interest, but in a given set there will be convergence to where this holds. This sets the stage for the LaSalle invariance principle. A fundamental concept in LaSalle's invariance principle is that for a state starting in a set $\boldsymbol{x}(t_0) \in M$, then M is an invariant set if $\boldsymbol{x}(t) \in M \ \forall t > t_0$. The LaSalle invariance principle states that given a region $\Omega_c = \{\boldsymbol{x} \in \mathbb{R}^n : \mathcal{V}(\boldsymbol{x}) \leq c\}$ for $c > 0$, define the subset $S_s = \{\boldsymbol{x} \in \Omega_c : \dot{\mathcal{V}}(\boldsymbol{x}) = 0\}$ with M being the largest invariant set in S_s. If $\boldsymbol{x}(t_0) \in \Omega_c$, then $\boldsymbol{x}(t) \to M$ as $t \to \infty$. Furthermore if the set M only contains the origin (which is the burden of proof), then the system is asymptotically stable over Ω_c. Given these definitions, control inputs can often be designed to satisfy the above constraints and make the system stable in given regions of interest. However to do this the system must be observable and controllable. A system is observable if the full state $\boldsymbol{x}$ can be determined in finite time from the outputs $\boldsymbol{z}$, in linear systems this is equivalent to the observability matrix being rank n (with n being the dimension of the state). A system is controllable if given two valid states $\boldsymbol{x}_0$, $\boldsymbol{x}_1$ there exist a time t_f such that for $\boldsymbol{x}(t_0) = \boldsymbol{x}_0$, with admissible inputs $\boldsymbol{u}(t)$, then $\boldsymbol{x}(t_f) = \boldsymbol{x}_1$ can be achieved.

Part II

Cooperative Transport

Chapter 4

Obstacle Modeling and Path Planning for Cooperative Transport

4.1 Introduction

The chapters in this section present considerations and methodologies for cooperatively transporting a load in a cluttered environment. Optimal Transport is defined here as taking the most efficient path to reach the goal and conserving energy during transport. When transporting on a flat terrain with simplistic robot kinematics, an efficient path usually correlates to the shortest path. During transport, energy is conserved when all applied forces are used to solely perform work on the object to transport it to the agreed upon goal. A centralized paradigm where there is a single governing controller can easily achieve optimal transport. This is done by

1. Selecting the desired, feasible goal.

2. Planning collision free trajectories for every robot and the carried load to the goal.

3. Leveraging knowledge of robot grasp locations to command every agent to apply forces that will mobilize to the goal and induce minimal internal stresses on the carried object.

In a decentralized paradigm, it is usually impossible to achieve the centralized performance without all agents having prior information as well as consensus.

In this section, decentralized robots and human collaborators are considered, and considerations for approaching optimal transport in the following way: first in Chapter 4 it is assumed that the feasible goal has been selected, and the goal is to find a collision free trajectory that optimizes a metric such as minimal distance. Then in Chapter 5 decentralized robot teammates will be considered, where each robot knows the optimal path from Chapter 4, but the grasp locations of other robot are unknown. Therefore to transport efficiently they must minimize the internal stresses in the carried object. Then finally in 6 human-robot transport is considered, where the final goal is known only to the human and the robot must observe the human and the environment and estimate the humans intended goal. Since the humans grasp location is known, a full pipeline would be the combination of all three chapters, as an optimal path can be planned to the humans intended goal, then given the grasp map, the robot knows what forces to apply to follow the optimal trajectory. The contributions of Chapter 4 is an efficient method of decomposing traversable free space for obtaining the optimal path. The contributions of Chapter 5 is determining the minimal information robots need to share with their immediate neighbors in order to reach consensus and minimize internal stresses. And the contributions of Chapter 6 is a novel approach to observe the human and environment and to predict on a small time horizon the human's intended goal pose. As the full pipeline implementation does not add novelty, it is left for future work.

4.2 Obstacle Modeling and Collision Checking

Obstacles around a robot can be detected with many types of sensors which can be classified into active and passive sensors. Active sensors project onto the environment and use this feedback to ascertain a quality about the environment with examples being sonar, laser range finder or even contact with the environment. Passive sensors detect the environment without disturbing it but measure an inherent quality, examples being cameras, temperature gauges, or an inertial measurement unit (which measures acceleration, angular velocity, magnetic field). Specifically for obstacle detection, common range sensors include laser range finders, time-of-flight sensors and stereo cameras. Points are registered in the sensor frame and transformed to the body or world frame with a measure of uncertainty. Obstacles from such a sensor are usually provided as a point cloud in the sensor frame.

There are many ways to represent the obstacle and the representation is usually driven by the particular application. Representation methods include as a raw point cloud, as occupancy voxels (with varying resolution), as polytopes (which contain the obstacle points) and statistically with mean and covariance of clusters of obstacle points. Given a representation, the objective is then to determine how to avoid collision. If the desired representation for the obstacles is a point cloud, then by modeling a polytope that encapsulates the robot, then only poses for the robot where the representing polytope does not include obstacle points are valid poses that avoid collision. The obstacles can be modeled as a set of polytopes whose dimensions are inflated subject to the pose of the robot so that it is possible to model the robot as a point. In this case, the planned path for the robot must be exterior to the obstacle polytopes, this is the method in mixed integer quadratic programming (MIQP) where points that are interior to obstacle polytopes are invalid in the optimizer solver [43]. If there are many distinct obstacles, this quickly becomes very expensive to compute a valid trajectory. Another approach is to represent obstacles as a point cloud, then characterize the traversable free space with overlapping convex regions [15, 29]. Given a method of characterizing and avoiding collision, the task becomes finding feasible trajectories that are collision free and satisfy a secondary objective such as minimal path length, allowing the robot to reach the goal in shorter time or by expending less power.

4.3 Generating Traversable Corridors

When both the goal pose is known along as well as a point cloud representation of the obstacles, then the traversable free space can be represented as a set of convex regions. This is useful, as an optimization problem can then be posed for a path that remains in the traversable free space and optimizes a secondary objective such as path length.

The free space decomposition can be performed in multiple ways, one common way is to define a polytope that contains the largest ellipsoid in the free region [15]. This method performs well for approximating the true free space, however, when the objective is traversing an environment to a known goal, it is not always necessary to have a complete representation of the entire free space, but only the effective corridor required to traverse collision free, and focusing on the region required for motion can reduce computational cost. This free space decomposition is proposed in [29] where the algorithm finds a seed path from a sample based planner or flood fill algorithm, then uses obstacles around the seed path chords to construct polytopes containing the chord segments. As these polytopes naturally overlap, this provides a set of overlapping convex regions from the start to goal pose. Defining the trajectory using the polynomial basis $\boldsymbol{t_v} = [1, t, \ldots, t^n]$ with coefficients $\boldsymbol{\beta} = [\boldsymbol{\beta}_x^T \; \boldsymbol{\beta}_y^T \; \boldsymbol{\beta}_\theta^T]^T$ so that the pose $\boldsymbol{p}_(t)$ of the robot in $SE(2)$ can be expressed as

$$\begin{aligned}
\boldsymbol{p}_{\mathrm{WR}}(t) &= \begin{bmatrix} x(t) & y(t) & 0 & 0 & 0 & \phi_z(t) \end{bmatrix}^T \\
x(t) &= \sum_i \boldsymbol{t_{v,i}} \beta_{x,i} \\
y(t) &= \sum_i \boldsymbol{t_{v,i}} \beta_{y,i} \\
\phi_z(t) &= \sum_i \boldsymbol{t_{v,i}} \beta\theta, i.
\end{aligned} \tag{4.1}$$

For $k-1$ chord segments there are $k-1$ independent polynomials and k boundary conditions. Individual polynomial segment time duration's are determined by the ratio of the initial length of the chord from the seed path: $T_{chrd} = \begin{bmatrix} T_{chrd,1} & \ldots & T_{chrd,k-1} \end{bmatrix}$, these times are then used to enforce equality constraints at polynomial connections. Within these polytope, a set of polynomials are found that satisfy the path objective function S_o for each i'th chord

$$\begin{aligned}
S_{o,i} &= \int_0^{T_{chrd,i}} ||\boldsymbol{p}_{\mathrm{WR}}^{(1)}(t)||^2 dt = \boldsymbol{\beta}_i^T P_{H,i}(T_{chrd,i}) \boldsymbol{\beta}_i \\
S_{o,total} &= \begin{bmatrix} \boldsymbol{\beta}_1 \\ \vdots \\ \boldsymbol{\beta}_{k-1} \end{bmatrix}^T \begin{bmatrix} P_{H,1}(T_{chrd,1}) & & \\ & \ddots & \\ & & P_{H,k-1}(T_{chrd,k-1}) \end{bmatrix} \begin{bmatrix} \boldsymbol{\beta}_1 \\ \vdots \\ \boldsymbol{\beta}_{k-1} \end{bmatrix} \\
&= \boldsymbol{\beta}^T P_H \boldsymbol{\beta}.
\end{aligned} \tag{4.2}$$

P_H is the Hessian of the total objective (summed over all k chords) $S_{o,total}$ with respect to polynomial coefficients and is the matrix form of the polynomial convolution (found with outer product of

$\boldsymbol{t_v}^T\boldsymbol{t_v}$ and put in block diagonal form). And the optimization problem becomes

$$\begin{aligned} &\min_{\boldsymbol{\beta}} \boldsymbol{\beta}^T P_H \boldsymbol{\beta} \\ &\textit{subj. to } A\boldsymbol{\beta} < \boldsymbol{b} \\ &\qquad E_{eq}\boldsymbol{\beta} = \boldsymbol{d}_{eq}. \end{aligned} \tag{4.3}$$

where the inequality $A\boldsymbol{\beta} < \boldsymbol{b}$ comes from the convex corridor generation and ensures the segment of the path remains interior to the respective convex corridor region. The equality constraint $E_{eq}\boldsymbol{\beta} = \boldsymbol{d}_{eq}$ ensures boundary conditions for the start and goal pose are met along with smoothness at connections of polynomials. This corridor generation algorithm is discussed in detail in [29]. For demonstration, this was implemented on the KUKA youBot which is a holonomic robot with four mecanum wheels. The states of the robot are $\begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix}^T = \begin{bmatrix} x & y & \theta \end{bmatrix}^T$, with input being the velocity vector $\boldsymbol{u} = \begin{bmatrix} \dot{x} & \dot{y} & \dot{\theta} \end{bmatrix}^T$. The kinematic equation of motion is therefore

$$\underbrace{\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{bmatrix}}_{\dot{\boldsymbol{x}}} = \underbrace{\begin{bmatrix} cos(x_3) & -sin(x_3) & 0 \\ sin(x_3) & cos(x_3) & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{g(\boldsymbol{x})} \boldsymbol{u} \tag{4.4}$$

Defining the change of coordinates to the error between the desired trajectory and current state

$$\boldsymbol{e}_x = \boldsymbol{x}_{des}(t) - \boldsymbol{x} \tag{4.5}$$

Then the combined control and optimization problem becomes

$$\begin{aligned}
\min_{\boldsymbol{u},\boldsymbol{\beta}} S &= \int_{t_0}^{t_f} \left(\boldsymbol{e}_x(t)^T Q_{\boldsymbol{e}} \boldsymbol{e}_x(t)\right) dt + \boldsymbol{\beta}^T P_H \boldsymbol{\beta} \\
\textit{subj. to } \dot{\boldsymbol{x}} &= g(\boldsymbol{x})\boldsymbol{u} \\
A\boldsymbol{\beta} &< \boldsymbol{b} \\
E_{eq}\boldsymbol{\beta} &= \boldsymbol{d}_{eq}.
\end{aligned} \tag{4.6}$$

where $Q_{\boldsymbol{e}}$ is a positive semi-definite matrix. The proposed controller is

$$\boldsymbol{u}(t) = g^{-1}(\boldsymbol{x})\left(\dot{\boldsymbol{x}}_{des}(t) + K_p \boldsymbol{e}_x(t)\right) \tag{4.7}$$

then as $g(\,\cdot\,)$ is a rotation matrix it is always invertible and the system is feedback linearizable and the kinematics becomes

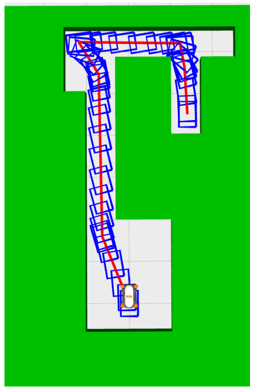
$$\begin{aligned}
\dot{\boldsymbol{e}}_x(t) &= \dot{\boldsymbol{x}}_{des}(t) - g(\boldsymbol{x})\left(g^{-1}(\boldsymbol{x})\left(\dot{\boldsymbol{x}}_{des}(t) + K_p \boldsymbol{e}_x(t)\right)\right) \\
&= -K_p \boldsymbol{e}_x(t)
\end{aligned} \tag{4.8}$$

which is asymptotically stable about the origin $\boldsymbol{e}_x(t) = 0$ when $K_p > 0$. If a trajectory is found that minimizes (4.3), then the control problem (4.6) is also minimized.

4.4 Results

This was implemented on a youBot base shown in Figure 4.1b. In Figure 4.2, red points represent seed path points, the green dashed path represents the desired trajectory, and blue represents the odometry using the control law presented for the experiment shown in Figure 4.1. In the Figure 4.2, the trajectories are over normalized time. This is scaled as in [43] to meet the desired trajectory traversal rate using the normed control time represented by $t_c = (t - t_0)/t_{scale}$ where t is the current time, t_0 is the start time, t_{scale} is the scaling.

To compare the proposed method to the leading methods in [39] and [15], all three methods were implemented in Robot Operating System (ROS) and their outputs examined for a single chord


(a)

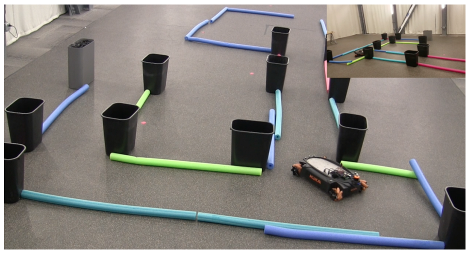
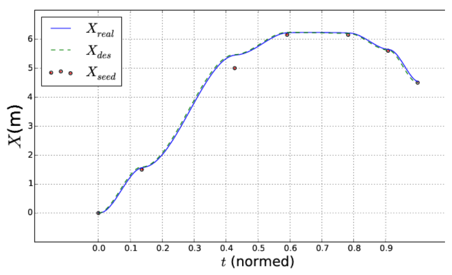

(b)

Figure 4.1: Implementation on KUKA youBot. Figure 4.1a shows the planned trajectory where the seedpath is shown in red and the optimal path for the convex hull of the robot is shown in blue. Figure 4.1b shows execution on the youBot platform.

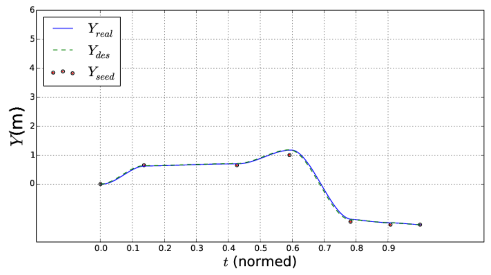
Table 4.1: Collision Free Corridor Generation Method Comparison: Free-space characterization for proposed method, and methods shown in [39] and [15]. These are statistics for the free space of the proposed method and Liu et al. method divided by free space of IRIS method respectively, producing a fractional relationship.

Trial (100 iter.)	Proposed method/IRIS	Liu/IRIS
5 obs.	0.84 $\pm$ 0.14	0.54 $\pm$ 0.21
10 obs.	0.77 $\pm$ 0.23	0.68 $\pm$ 0.18
30 obs.	0.81 $\pm$ 0.20	0.97 $\pm$ 0.23
50 obs.	0.82 $\pm$ 0.23	1.03 $\pm$ 0.22

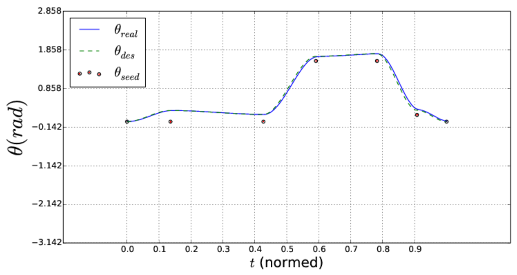
defined from start $x = 0$, $y = -0.3$ to goal $x = 0$, $y = 0.3$. The simulation units are consistent for comparison, and are considered to be meters. For this chord obstacles are generated within the world bounds of $x_w = \pm 1m$, $y_w = \pm 1m$ and discretized the space with step size of $0.02m$ in both x, y. This is visualized in ROS-rviz, shown in Figure 4.3. Figure 4.3a shows the IRIS methods' free, convex space, Figure 4.3b shows the proposed methods free convex space for this chord, and Figure 4.3c shows the free space generated by the Liu et al. method in [39]. It is important to note that the Liu et al. method performs ray tracing, and subdivides a chord into sequential convex regions defined by co-linear ray intersections. Hence while the Liu et al. method can be compared in terms of free space, it cannot determine the convex space around a single chord as in the proposed method and IRIS method [15]. In Table 4.1 the free space is calculated for the system shown in Figure 4.3 for 10, 30 and 50 obstacles each for 100 iterations. By dividing each respective calculated free



(a) X position vs normalized time



(b) Y position vs normalized time

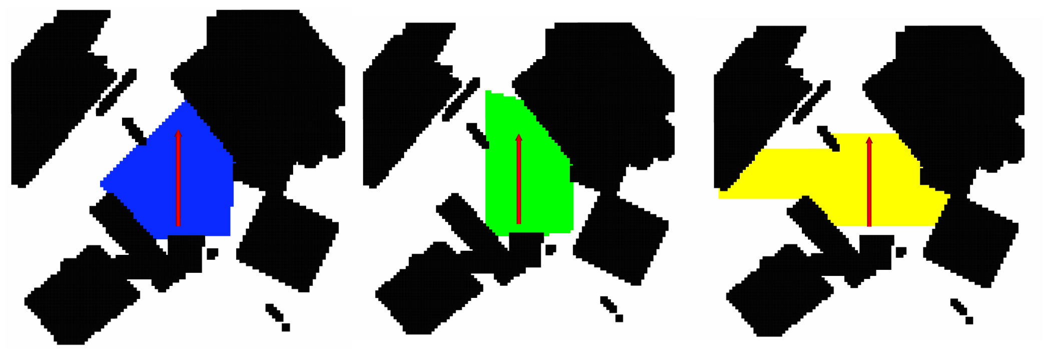


(c) θ position vs normalized time

Figure 4.2: Smooth trajectory generation where red points show the seed path, green dashed lines are the desired trajectory and blue lines represent the actual odometry.

space by the IRIS method, a scale free relation is obtained that allows for comparison of free space between all methods, and convex space between IRIS and the proposed method.

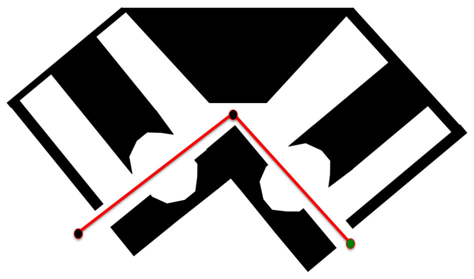
Table 4.1 shows the free space comparison. It is important to note that the world size remains constant while the number of obstacles increases for each trial (100 trials for each number of obstacles). The proposed method is consistently about 81% (with less than 23% standard deviation) of the free convex space calculated by IRIS. The Liu et al. method computes less free space than the proposed and IRIS methods in open environments with few obstacles, but as the environment becomes more cluttered, the ray-tracing feature of the Liu et al. method allows it to find more free space. It is important to note that the free space found in the Liu et al. method in these instances does not translate to usable convex free space required for optimization (as shown in Figure 4.3c). In addition, the Liu et al. method's ray tracing makes it inherently sensitive to world discretization and dimensionality. The IRIS method finds the maximum convex polytope that can contain the largest ellipsoid. As the proposed method formulation is to expand a polytope around the chord, there are instances when this convex volume will exceed the IRIS method (as it solves a similar



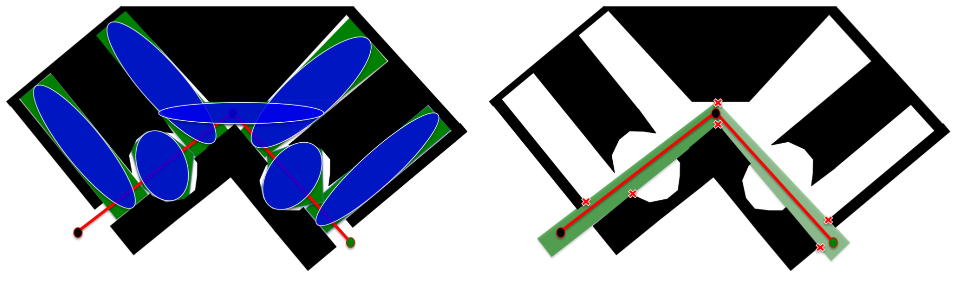
(a) IRIS SDP method (b) Proposed method (c) Liu et al. method

Figure 4.3: General method comparison of free space: single chord indicated by red arrow, 30 generated obstacles and free space characterized by each method.

but different problem). It must be noted that the key difference between the proposed method and the IRIS method are that they solve different problems, with the main goal of the proposed method being to find the maximum usable free space around a given chord and the goal of IRIS to find the polytope associated with maximum volume ellipsoid in the region. This leads to the apparent advantage of computation reduction in the proposed method. Consider Figure 4.4c, in Figure 4.4a



(a) Seed path in cluttered environment



(b) IRIS Method [15] (c) Proposed method [29]

Figure 4.4: Given a seed path, the IRIS method requires more computation to calculate the effective, usable free-space than the proposed method.

a seed path has been generated in a cluttered environment. The IRIS method shown in Figure 4.4b calculates the free space throughout the environment, and requires seven computation cycles in this example. However, the proposed method shown in Figure 4.4c only requires two computation cycles. So while the proposed method does not identify all of the possible free space, it does locate all of the useful free space for the task of trajectory generation. In this analysis the proposed methods effectiveness is demonstrated for the given application, and is compared to similar methods for free space decomposition.

Chapter 5

Robot-Robot Cooperative Transport

5.1 Introduction

In this chapter the scenario of multiple, decentralized robots cooperatively carrying a load is investigated. It is assumed that the goal pose is known and a collision free trajectory is determined and known to all robotic agents as in Chapter 4. It is also assumed that every robot is capable of measuring the state of the load and the resultant applied wrench assuming a quasi-static paradigm (inertial forces are zero, and applied forces are countered by friction for the system to traverse at constant velocity). Each robot also knows where it is grasping the object, however while the robot is able to communicate with adjacent robots within a communication radius, the grasp points of other agents is unknown, and as the number of robots greatly increases this information may become intractable for some robotic platforms. Hence, the control problem becomes transporting the object along the specified trajectory while minimizing interaction forces which do not contribute to motion of the system.

5.2 Force Distribution in Cooperative Transport

Consider N robots grasping a load at unique points $\boldsymbol{r}$ in the inertial frame. If each agent is capable of producing a force (but not a torque) at grasp points then the wrench on the body (force and torque) is defined as

$$\begin{aligned}
\boldsymbol{w} &= \sum_i \begin{bmatrix} I \\ \boldsymbol{r}_i^{[\times]} \end{bmatrix} \boldsymbol{f}_i \\
&= \sum_i Gi \boldsymbol{f}_i \qquad (5.1)
\end{aligned}$$

The matrix Gi will be referred to as the grasp map as it describes the contribution to net wrench the i'th robot can provide. Two robots will produce internal stresses on the load if there are components of their forces that do not contribute to the net wrench. These are denoted as interaction forces, described for two robots i, j as

$$f_{ij} = (\boldsymbol{f}_i - \boldsymbol{f}_j)^T \hat{\boldsymbol{r}_{ij}} \qquad (5.2)$$

where

$$\hat{\boldsymbol{r}_{ij}} = \frac{(\boldsymbol{r}_i - \boldsymbol{r}_j)}{||\,(\boldsymbol{r}_i - \boldsymbol{r}_j)\,||} \qquad (5.3)$$

If the objective is to minimize interaction forces, then this can be achieved using a common solution vector $\boldsymbol{\beta}$ for all robots

$$\boldsymbol{f}_i(\boldsymbol{\beta}) = Gi^T \boldsymbol{\beta} \ \forall i \qquad (5.4)$$

Substituting this into (5.2) guarantees no interaction forces because

$$\begin{aligned}
f_{ij}(\boldsymbol{\beta}) &= (\boldsymbol{f}_i(\boldsymbol{\beta}) - \boldsymbol{f}_j(\boldsymbol{\beta}))^T \frac{(\boldsymbol{r}_i - \boldsymbol{r}_j)}{||\,(\boldsymbol{r}_i - \boldsymbol{r}_j)\,||} \\
&= \left(\boldsymbol{\beta} - \boldsymbol{r}_i^{[\times]}\boldsymbol{\beta} - \boldsymbol{\beta} + \boldsymbol{r}_j^{[\times]}\boldsymbol{\beta}\right)^T \frac{(\boldsymbol{r}_i - \boldsymbol{r}_j)}{||\,(\boldsymbol{r}_i - \boldsymbol{r}_j)\,||} \\
&= \frac{\boldsymbol{\beta}^T}{||\,(\boldsymbol{r}_i - \boldsymbol{r}_j)\,||} \left(\boldsymbol{r}_i^{[\times]}\boldsymbol{r}_i - \boldsymbol{r}_j^{[\times]}\boldsymbol{r}_i - \boldsymbol{r}_i^{[\times]}\boldsymbol{r}_j + \boldsymbol{r}_j^{[\times]}\boldsymbol{r}_j\right) \\
&= \frac{\boldsymbol{\beta}^T}{||\,(\boldsymbol{r}_i - \boldsymbol{r}_j)\,||} \left(\boldsymbol{r}_i^{[\times]}\boldsymbol{r}_j - \boldsymbol{r}_i^{[\times]}\boldsymbol{r}_j\right) \\
&= 0 \qquad (5.5)
\end{aligned}$$

So for optimal transport, the optimization problem with respect to interaction forces can be stated as

$$\min_{\boldsymbol{f}_i} S_{int} = \int_{t_0}^{t_f} f_{ij}^2(t)dt \ \forall i, j \tag{5.6}$$

and consensus of the solution vector $\boldsymbol{\beta}$ in (5.4) minimizes this objective.

5.3 Interaction Force Minimization Through Consensus

For robots to reach consensus on their solution vector to minimize interaction forces, they must communicate with their neighbors, and through averaging, reach consensus. It is assumed that robots are only able to communicate with neighbors that are within a radius r_{comm}. The communication structure can be represented using a Laplacian matrix:

$$[\mathbb{L}_{i,j} = \begin{cases} d_i & \text{if } i = j \\ -1 & ||\boldsymbol{r}_i - \boldsymbol{r}_j|| \leq r_{comm} \\ 0 & ||\boldsymbol{r}_i - \boldsymbol{r}_j|| > r_{comm} \end{cases}$$

Where d_i are the number of neighbors for the i'th robot. The Laplacian $\mathbb{L}$ of graph $\mathbb{G}$ is defined as $\mathbb{L} = \mathbb{D} - \mathbb{A}$, and is a symmetric matrix (in an undirected graph). $\mathbb{D}$ is a diagonal matrix populated by d_i, and $\mathbb{A}$ is an adjacency matrix indicating neighbors. If the graph is connected (starting on any node it is possible to reach any other node over adjacent edges), then the first eigenvalue of the Laplacian is zero and the rest are positive. For Laplacian averaging, the discrete consensus algorithm is

$$\boldsymbol{x}[s+1] = \boldsymbol{x}[s] - \gamma \mathbb{L}\, \boldsymbol{x}[s]. \tag{5.7}$$

Based on the choice of γ this algorithm converges to the average after m iterations for N robots

$$\lim_{m\to\infty} \boldsymbol{x}[s+m] = \frac{\mathbf{1}\mathbf{1}^T}{N}\boldsymbol{x}[s]. \tag{5.8}$$

In [30] a discrete control law is proposed, but in this analysis a continuous control law will be developed.

If it is asserted that consensus has been reached for the solution vector in (5.8), then the net wrench can be expressed as

$$\begin{aligned} \boldsymbol{w}(t) &= \sum_{i}^{N} GiGi^{T}\boldsymbol{\beta} \\ &= G\boldsymbol{\beta} \end{aligned} \tag{5.9}$$

for N robots, where the matrix G combines the entire grasp map. For simplicity the body fixed frame is considered so that G is constant. Differentiating this provides the change in wrench required to track a desired trajectory and represents the dynamics of the system

$$\dot{\boldsymbol{w}}(t) = G\dot{\boldsymbol{\beta}}. \tag{5.10}$$

Defining the tracking error to be

$$\boldsymbol{e}_{\boldsymbol{w}}(t) = \boldsymbol{w}_{des}(t) - \boldsymbol{w}(t) \tag{5.11}$$

and the control input to be

$$\boldsymbol{u}(t) = \dot{\boldsymbol{\beta}} \tag{5.12}$$

the objective is to minimize the following functional

$$\begin{aligned} \min_{\boldsymbol{u}_i} S &= \int_{t_0}^{t_f} \left(\boldsymbol{e}_{\boldsymbol{w}}(t)^T Q \boldsymbol{e}_{\boldsymbol{w}}(t) + f_{ij}^2(t) \right) dt \\ \textit{subj. to } & \boldsymbol{w}(t) = G\boldsymbol{u}(t), \end{aligned} \tag{5.13}$$

where the consensus step in (5.8) minimizes the interaction force component and Q is a positive semi-definite matrix. It is now asserted that the desired wrench $\boldsymbol{w}_{des}(t)$ is feasible meaning that it remains in the column space of G, and by extension that the error $\boldsymbol{e}_{\boldsymbol{w}}(t)$ also remains feasible. Based on this assertion, using a change of coordinates by differentiating (5.11) the dynamics can be expressed by

$$\begin{aligned}\dot{\boldsymbol{e}}_{\boldsymbol{w}}(t) &= \dot{\boldsymbol{w}}_{des}(t) - \dot{\boldsymbol{w}}(t) \\ &= G\boldsymbol{\beta}_{des}(t) - G\boldsymbol{u}(t) \end{aligned} \tag{5.14}$$

By selecting the control policy

$$\boldsymbol{u}(t) = \boldsymbol{\beta}_{des}(t) + G^{+}K_p\boldsymbol{e}_{\boldsymbol{w}}(t) \tag{5.15}$$

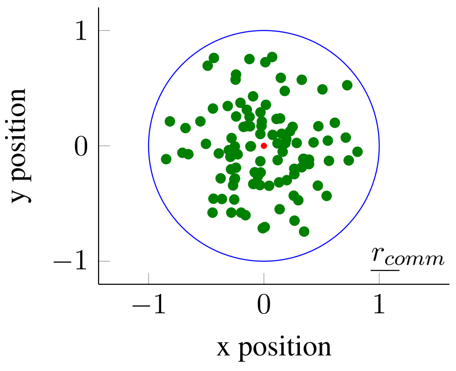
substituting (5.15) into (5.14) this becomes

$$\dot{\boldsymbol{e}}_{\boldsymbol{w}}(t) = -GG^{+}K_p\boldsymbol{e}_{\boldsymbol{w}}(t) \tag{5.16}$$

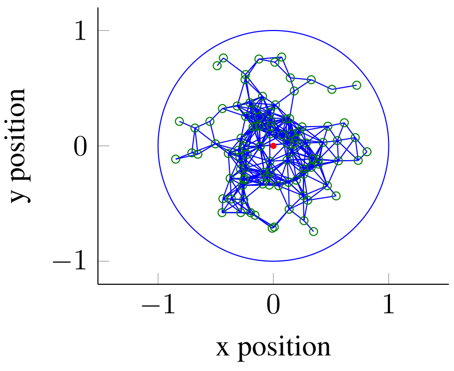
which for $K_p > 0$ and valid feasibility assertion (meaning $\boldsymbol{e}_{\boldsymbol{w}}$ is in the column space of G, whose eigenvalues are greater than or equal to zero), is asymptotically stable.

5.4 Results

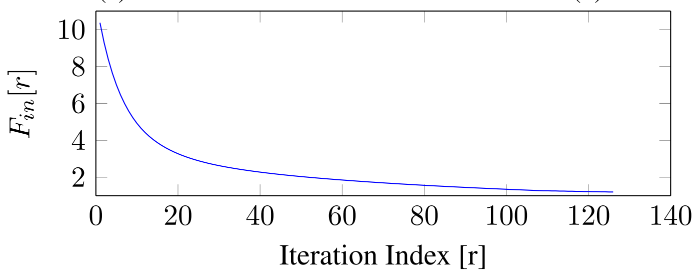
Figure 5.1 shows a representative example of 100 robots with unique random contact locations executing a desired wrench of $\boldsymbol{w}_{des} = \begin{bmatrix} 1, & 1, & 1 \end{bmatrix}^T$. This is representative as contact points were randomly selected without replacement within the object boundary. Notice in Figs. 5.1e and 5.1d that consensus among the total number of robots takes about one hundred iterations; however, after consensus is reached, since all the estimates of $\boldsymbol{\beta}_i$ are the same and the graph does not change, the system converges to the desired wrench quickly (noted by a decrease in wrench error $\|\boldsymbol{w}_{err}[s]\| = \|\boldsymbol{w}_{des}[s] - \boldsymbol{w}_{act}[s]\|$). Notice in Fig. 5.1c that the sum of the absolute value of the interaction forces $\boldsymbol{f}_{in}[s] = \sum_{i,j} |(\boldsymbol{f}_i[s] - \boldsymbol{f}_j[s]) \cdot (\boldsymbol{r}_i - \boldsymbol{r}_j)|$ is steadily decreasing as a result of the values of $\boldsymbol{\beta}_i$ converging to their average $\boldsymbol{\beta}_{avg}$, as seen by the decrease in the first standard deviation σ_1. The convergence graph in Fig. 5.1e shows the average of the robots estimated solution vector $\boldsymbol{\beta}_i$ as well as the first standard deviation of these estimates σ_1. This algorithm is robust to communication failure as long as the set of graph topologies during the communication failure constitute a connected graph [30].



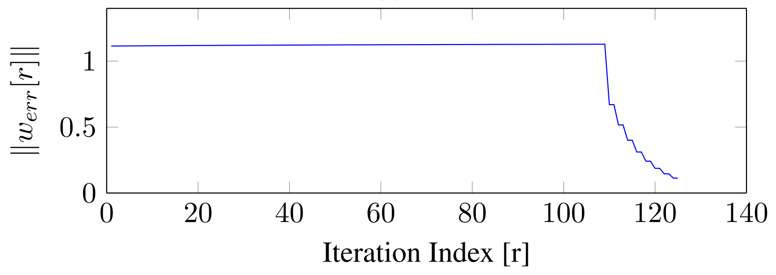
(a)



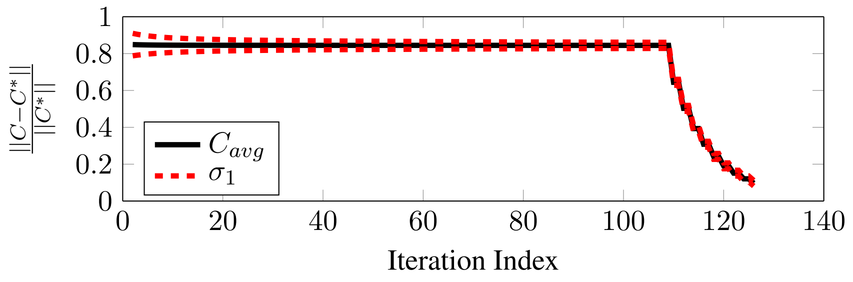
(b)



(c)



(d)



(e)

Figure 5.1: Robot-Robot Cooperative Carrying: 100 robots executing $\boldsymbol{w}_{des} = [1,\,1,\,1]^T$, with a static communication structure and convergence rate.

Chapter 6

Human-Robot Cooperative Transport

6.1 Introduction

In this chapter, cooperative transport between a human leader and robot follower is considered. The novelty and contribution in the proposed approach is the use of human and environment observations to predict the human's intended goal pose on a small time horizon. It is an added constraint is that the human cannot explicitly communicate with the robot (verbally, digitally, or using predefined gestures). However it is assumed that

1. The human has an intended pose for the carried object.
2. The human wants the system to avoid collisions with surrounding obstacles.
3. The human wants to transport in an energy efficient manner (which may include but is not limited to traveling the shortest distance to reach target pose).
4. The obstacles in the environment are static.
5. The human-robot pair will carry the object such that the humans face is visible by the robot, along with the humans legs. And in addition that the human applied wrench can be sensed. This is shown in Figure 6.1 and Figure 6.2.
6. The obstacles in the environment can be sensed by the robot and the robot is able to measure the carried objects odometry in the inertial reference frame.

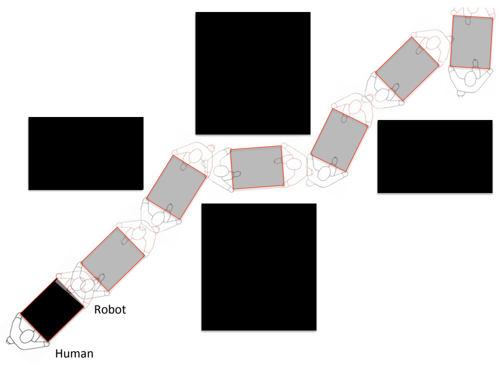


Figure 6.1: Human Robot Carrying Problem: The goal is to allow an autonomous robot to cooperatively carry a load through a cluttered environment collision free, with a human counterpart with the added constraint that explicit communication is not viable. The robot must transport efficiently by estimated the humans intended motion in the context of the environment.

Leveraging the contributions of Chapter 4 and Chapter 5 it is asserted that once the intended goal pose is estimated, the robot is able to plan a collision free trajectory to the goal and by observing the grasp location of the human, apply an appropriate wrench that will minimize internal stresses on the carried object and conserve energy.

In this chapter, what the robot can observe from the human and environment will first be presented. And based on these observations, a physics based reasoning will be proposed for how each of these observations may predict the humans intended pose. Second, considerations for modeling of human intent based on these observations will then be discussed. And finally positive preliminary results will be shown as an initial attempt at modeling the human intent on a small time horizon.

6.2 Human and Environment Sensing for Cooperative Transport

Consider the scenario when a robot carries a load with a human collaborator as shown in Figure 6.1. When the final destination is known by both the robot and human, a feasible trajectory can be calculated and if the grasp locations are known by all collaborators then the desired motion can be performed. However, if only the human leader knows the final target pose, then it becomes difficult for the robot to be an effective collaborator. One method to mitigate this is to allow the robot to simply become compliant, if the human leader applies forces to the carried load the robot can move in manner that then minimizes the internal stress sensed. This method has been used widely in robotics for decades, but has the fundamental drawback that there must be internal stresses in the carried load in order to signal the robot to move in a particular direction for transport [81], [3], [63].

To mitigate these internal stresses the robot must be able to estimate the intended pose before the human is required to send the implicit signal of internal stresses through the carried object. If the context of the object being carried is known within the given environment then it is possible for the robot to plan long time horizon trajectories based on these affordances. An example of this would be carrying a sofa through a house to the living room to be placed against a designated wall. In this example, if the robot has a floor-plan of the house a collision free trajectory can be calculated to the intended pose. When considering the deployable robotic assistant, it is fair to assume that without prior experience the generic affordance of the carried object correlated to a given floor-plan may not exist or multiple solutions may exist.

Hence, for long term deployment, if the robot can continuously and accurately approximate the human leaders intended goal pose on a local time horizon (a few seconds into the future), then the entire transport task can become efficient. The challenge becomes determining what can be observed by the robot and is useful for estimating the human leaders intended pose. Given the transport scenario shown in Figure 6.1, consider a local region around the human and robot collaborators. It is assumed that the robot is capable of sensing the immediate obstacles, the orientation of the humans head, the motion of the humans legs, the wrench applied by the human as well as the odometry of the carried load as shown in Figure 6.2. For a small time horizon it is asserted that these quantities in whole or in part measured at time t can be used to predict the future pose of the object (whose frame is O) at a look ahead time horizon $t + t_h$. The assertion is based on the following intuitive underlying physics models:

1. **System Twist:** The system twist $\dot{\boldsymbol{p}}_{\mathrm{W}sys}$ obtained from the odometry can be integrated once to solve for the expected pose if it is assumed that the velocity will remain constant.

2. **Human Applied Wrench:** The applied wrench $\boldsymbol{w}$ which consists of applied forces and torques are a good indicator of intended acceleration. This can therefore be integrated twice to obtain the expected pose.

3. **Human Head Orientation:** The orientation of the human head $\boldsymbol{h}$ may indicate a desired

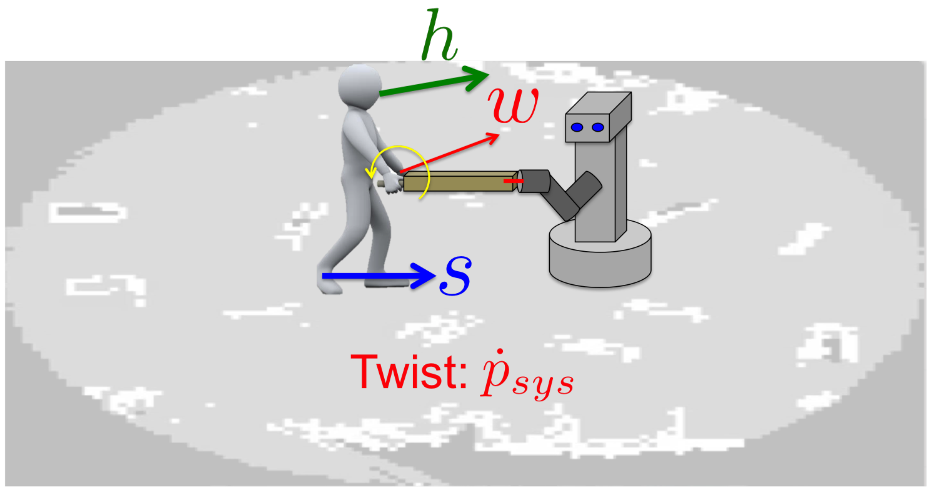


Figure 6.2: Human Intent Observation: It is asserted that the human and robot will transport facing each other. It is also asserted that the robot is capable of sensing the human leader's head orientation $\boldsymbol{h}$, step vector $\boldsymbol{s}$, applied wrench $\boldsymbol{w}$ (force and torque), the twist of the carried load $\dot{\boldsymbol{p}}_{\mathrm{W}sys} := \dot{\boldsymbol{p}}_{\mathrm{WO}}$ (for carried load frame O) and the surrounding obstacles.

direction of motion. As it may be difficult to extrapolate the intended orientation, the position may be estimated. It is important to note that the head orientation by itself does not correlate to the expected pose, however when normalized and scaled by the magnitude of the current planar velocity of the carried object, it can then be treated as an expected twist and integrated once for expected pose. It is also important to note that due to the coupling with the system twist, if the person is walking backward then it is safe to assume that the head orientation is not viable if the human cannot observe the intended pose. This is equivalent to the head orientation vector being in the same half-space as the twist planar vector.

4. **Human Step Vector:** The step vector $\boldsymbol{s}$ indicates the motion of the human and can be obtained by extracting consecutive locations of the humans center of location. This is done by detecting the leg locations and defining the centroid of the convex volume which contains the legs as the center of location. It is important to note that across different users the magnitude of the step vector may not be a direct indicator of the expected pose of the carried object. For instance, a shorter person's step may be an equivalent indicator of desired motion as a larger step taken by a taller person. It is assumed that the step regardless of magnitude, indicates the desired motion. Therefore, if the step vector is normalized and scaled by the magnitude of the

current planar velocity of the carried object, it can then be treated as an expected twist vector and integrated once for expected pose. This is expected to allow for consistent performance across users of different heights and therefore different average step sizes. It is important to note that if the step vector is not in the same half-space of the twist planar vector then it may not be useful. Intuitively, if the step vector is correlated with twist and is not in the same half-space then it is not a feasible predictor. If correlated to acceleration it may be useful, but its correlation to twist will be considered for now.

5. **Local Obstacles**: The local obstacles serve as motion constraints. If it is assumed that the human leader does not which to collide with obstacles, then if any of the above integration's would cause collision then it must be adjusted or negated from consideration.

The xy-planar vectors for the head and step are denoted as $\boldsymbol{h}$ and $\boldsymbol{s}$ respectively. The effective head vector $\tilde{\boldsymbol{h}}$ is defined based on the planar twist of the object whose frame is O expressed in the world W $\dot{\boldsymbol{p}}_{\mathrm{WO},xy}$ as

$$\tilde{\boldsymbol{h}} = \begin{cases} \frac{||\dot{\boldsymbol{p}}_{\mathrm{WO},xy}||}{||\boldsymbol{h}||}\boldsymbol{h} & \text{if } \boldsymbol{h} \cdot \dot{\boldsymbol{p}}_{\mathrm{WO},xy} > 0 \\ \boldsymbol{0} & \text{if } \boldsymbol{h} \cdot \dot{\boldsymbol{p}}_{\mathrm{WO},xy} \leq 0. \end{cases} \tag{6.1}$$

Likewise, the effective step vector $\tilde{\boldsymbol{s}}$ is defined as

$$\tilde{\boldsymbol{s}} = \begin{cases} \frac{||\dot{\boldsymbol{p}}_{\mathrm{WO},xy}||}{||\boldsymbol{s}||}\boldsymbol{s} & \text{if } \boldsymbol{s} \cdot \dot{\boldsymbol{p}}_{\mathrm{WO},xy} > 0 \\ \boldsymbol{0} & \text{if } \boldsymbol{s} \cdot \dot{\boldsymbol{p}}_{\mathrm{WO},xy} \leq 0. \end{cases} \tag{6.2}$$

The next challenge becomes appropriately combining knowledge from each of the sensing modalities mentioned. To learn how humans leverage this information, data was collected between two human teammates transporting a sensor capable of obtaining the above sensing quantities. The sensor is shown in Figure 6.3, and consists of a RGB-D Kinect sensor to measure the human leader's head orientation, a laser range finder to obtain the leader's location, strain gauges to measure the planar forces applied by the human leader and markers used to track odometry using a motion capture system. To obtain the local map, a Fetch mobile manipulator was used to obtain an obstacle map in the motion capture system's frame as shown in Figure 6.4a. And to collect data, a human leader

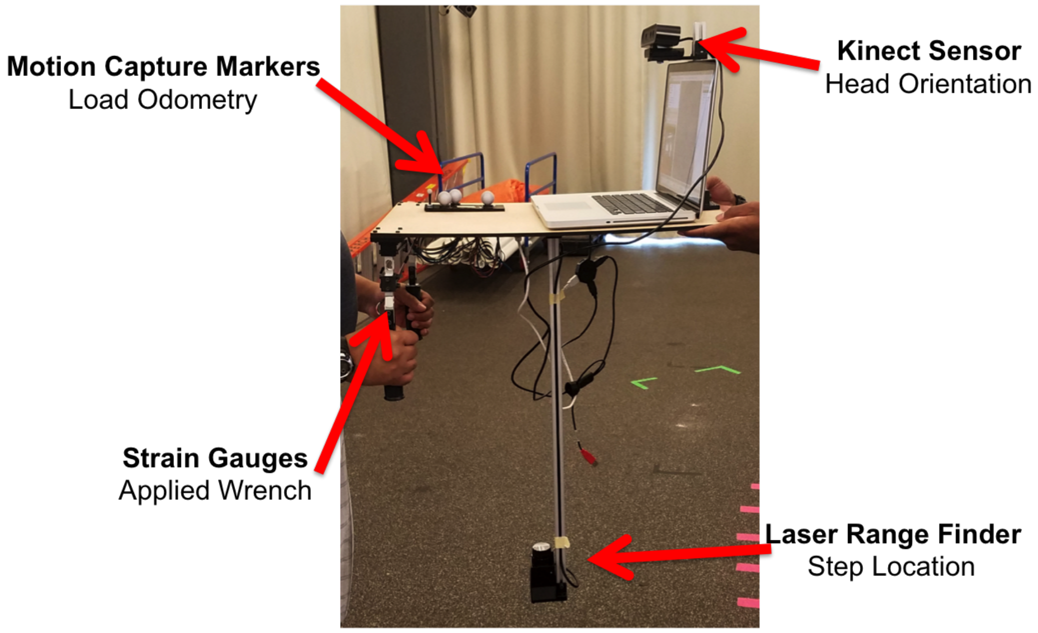


Figure 6.3: Human Intent Sensor Diagram: The sensor consists of a RGB-D Kinect sensor used to measure the human leader's head orientation, a laser range finder to obtain the leader's location and extract the step vector, strain gauges to measure the planar applied forces, and markers used to obtain odometry using a motion capture system.

and follower traversed the obstacle course with the constraint that the human leader was unable to verbally indicate or gesture the intended pose to the human follower (as shown in Figure 6.4b).

Data was collected over six different maps shown in Figure 6.5. For each trial the human leader selected multiple destinations and the human follower observed the leader while attempting to minimize internal stresses and avoid obstacles. The task becomes determining the appropriate way to combine the sensed quantities in order to effectively predict the human leaders intended goal pose.

6.3 Modeling Human Intent for Cooperative Transport

In order to predict the expected pose on a finite time horizon the challenge of correctly combining the measured quantities must be addressed. One approach is to assert independence between the quantities. In certain instances this may be effective, for example, with few obstacles the human leader may choose a low curvature trajectory with constant velocity making integration of the twist a valid indicator. Similarly, in an open space, the applied wrench may be a sufficient indicator of

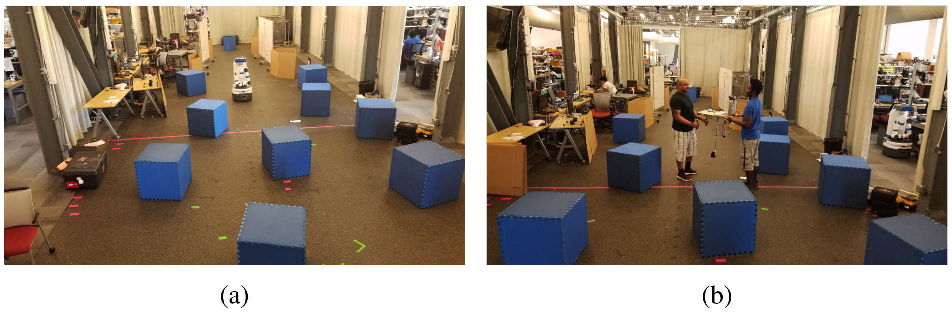


Figure 6.4: Human Intent Data Collection: The Fetch mobile manipulator is used to construct a map of the environment. Then a human leader and human follower then traverse the environment without explicit communication (verbal, gesture). By observing the human leader and the environment, the human follower predicts the leaders intended pose.

acceleration and be a sufficient predictor for the intended pose. Likewise for the effective head and step vector in an open space, they may be good indicators of the intended pose. However, consider the end of a hallway with two sharp turns, in this instance the human leader may look in one direction (the intended path) and apply a wrench in the opposite direction (to rotate the object). In this and similar examples the independent models are may be incapable of providing a complete description of the intended motion. It is also important to note the influence the surrounding obstacles have on the properties of these models. For example, in an open space, the head orientation may be a valid indicator of where the human leader intends to move, but in a tight hallway if the human leader looks toward a wall it is assumed that the head orientation is meaningless as collisions are to be avoided.

So it is apparent that coupling these models can be advantageous in many scenarios, however determining the most effective method of combining them is not straight forward. Consider a linear model that combines the integrated pose from each of the measured quantities, if weights are used to combine them, finding valid weights that improve the combined models performance can be posed as system identification problem. An inherent challenge however is determining the best combination, if the underlying model is not guaranteed to be a linear combination, then the model variations must also be considered. This consideration creates a huge search space to determine the appropriate model. One approach to solve such problems is to use genetic algorithms in which many model forms are generated, improved and evaluated to hopefully find the model with best performance. A limitation of such an approach is that if an inadequate pool of models is generated then performance will always be poor. This can easily occur when the engineer selects the base components that compose the set of all models and if the required true model base components are not present the model selection process may never converge to a valid selection.

This particular problem of utilizing the quantities sensed in Figure 6.2, the contribution of the local map should not be understated as it may change the model behavior for the other quantities. Once approach to solve this is to assume that the human follower performs using an adequate and possibly optimal internal model. With this assertion, the goal becomes modeling the human followers performance to reproduce their prediction accuracy. For a small time horizon, it is asserted that a Markov Process (MP) sufficiently describes the underlying model, meaning quantities in Figure 6.2 measured at time t are sufficient to predict the pose at $t + t_h$ without the history of quantities before time t. For longer time horizons this assertion is expected to quickly break down, hence this is restricted to small time horizons (in this work this is less than two seconds).

Given the supervised data collected in Section 6.2 a discriminative model framework can be leveraged to approximate the underlying model. Two modeling methods to be consider are a Gaussian process and a neural network. For the discriminative model framework the input state space is the measured quantities and the output as the intended pose. The advantage of using a Gaussian process is that training performed by constructing the information matrix and can therefore be performed quickly while data is collected. A disadvantage of using the Gaussian process is when the input state space manifold is highly nonlinear. In this case more training points are required near the non-linearity to adequately represent the manifold and therefore the information matrix increases proportionally. One way to overcome this is to use sparsification which only keeps training points represent manifold curvature beyond the specified set point. Another way to overcome this is to use a local Gaussian process which takes effect in specified regions of the state space. When the manifold is unknown and possibly very nonlinear both of these approaches may still break down with computational consequences. A neural network is a better approach in such a scenario, as the number of model representation components in the computational graph remains constant for a selected neural net architecture. Another advantage of a neural network is its inherent ability to approximate any bounded, continuous function provided an adequate architecture. This advantage is formalized in the Universal Approximation Theorem [23].

With these considerations, the proposed approach uses a deep neural network shown in Figure 6.6 using supervised training from data collected in Section 6.2. First, all of the measured quantities and local map are transformed into the current body fixed frame of the carried object O at time t. In addition, the future pose at look ahead time $t + t_h$ is also transformed into the body fixed frame O at time t. Intuitively, the primary architecture is described by a set of convolutional layers that scan the local map extracting the notion of constraints, the effective head and step pass together through a set of fully connected networks as it assumed that they both are indicators of the expected planar position. Finally, the output from the map, the effective head and step are combined with the applied wrench and carried object twist through a set of fully connected layers in order extract the inherent coupling for the prediction of the intended pose. Leaky ReLU activation functions are used for all nodes.

For supervised training, given the MP assumption, each carrying trial can be subdivided with measured quantities at time t and true pose at time $t + t_h$ creating a training set. When training, batches are randomly selected from all carrying trials with no ordering, allowing for the extraction of the model under the MP assumption. The error function was designed to penalize the $SE(2)$ error with position in units of centimeters and orientation in degrees. By squaring the error, the network would approach the true model.

Two back-propagation methods were considered, first was error in the final predicted $SE(2)$ pose as well as positional error $p_{sys,xy}(t + t_h)$ with the output from the fully connected networks connected to the effective head and step vectors. The hypothesis was that the effective head and step vectors correlated directly to the expected planar position. This method of back-propagation was found to not converge even with further modification in the neural net architecture. This led to the conclusion that while the head and step may indicate the expected planar position it is not sufficient to enforce assertion. The second back-propagation method tested was to just use the supervised error with the final $SE(2)$ pose. It was determined that this back propagation method led to convergence.

Many renditions of the internal architecture were tested with varying convolutional layer depths and channels, varying depth in fully connected layers for both the effective head and step as well as the final fully connected layers which combined all inputs. General trends indicated that sufficient depth was required for adequate model generalization, and sufficient width in fully connected layers improved model approximation. The final neural net architecture design selected is shown in Figure 6.6. All of the intent quantities are transformed into the local body fixed frame O, along with the future pose obtained at $t + t_h$. The deep network shown in Figure 6.6 has two convolutional layers that operate on the local map, the output of which is passed through a fully connected layer. The deep network passes the effective head and step vectors through a set of fully connected layers. Then the network combines the output from the effective head and step along with the applied wrench and body twist in a set of fully connected layers. Finally, the output from the effective head and step, wrench and body twist are all combined with the output from the map and passed through multiple fully connected layers to estimate the expected $SE(2)$ pose of the carried load for a given time horizon t_h. The guiding rational behind the construction of this network is that similar information is first combined and passed through layers that model pertinent features, then later, this output is combined with more dissimilar information in order to obtain the final desired output. For instance, the head and the step both intuitively indicate future position (not orientation). Whereas the applied wrench is a good indicator of acceleration. The system twist is directly useful for integration for expected pose. The local map is a good indicator of immediate constraints, if any other prediction would cause collision this would be infeasible. The convolutional and subsequent fully connected layers serve to extract this notion of constraint. The information from all of these quantities is then mapped through the multiple fully connected layers to a final output of expected pose for a given time horizon. While the best performing architecture is shown in Figure 6.6, competing architectures were examined. Most notably, an architecture that made the final error function a combination of final predicted pose error summed with the position error between the true position and the output of the head, step vectors fully connected layers was attempted with the hypothesis that the head and step vectors directly correlate to the expected position. Error convergence did not occur for this architecture, and the reasoning is that while the head and step vectors are good indicators of position, across all scenarios it is imperative that the surrounding obstacles and other intention quantities be considered in unison to predict the accurate pose. This is exemplified in the final architecture Figure 6.6.

6.4 Results

To estimate the human intended pose on a small time horizon from measured quantities shown in Figure 6.2, the final selected neural net architecture is shown in Figure 6.6. Independent models were trained for for time horizons of $0.5s$, $1.0s$, $1.5s$ and $2.0s$. The data size for each time horizon is shown in Table 6.1, and the hold out set (training data never touched and used to truly evaluate model performance) is 15% of the data size.

Table 6.1: **Human Intent Training:** Using the MP assumption for each time horizon t_h, the data size is shown and the hold out percentage is 15% of the data size.

Time horizon (sec)	Data size
0.5	2912
1.0	2902
1.5	2890
2.0	2878

Figure 6.7 shows the model error during training on the hold out set for the positional error in centimeters squared $||\boldsymbol{e}_{xy}||^2$ on a semi-log graph. Figure 6.8 shows the model error during training on the hold out set for the angular error in degrees squared $\boldsymbol{e}_{\theta}^2$ on a semi-log graph. Finally, Figure 6.9 shows the combined error of Figure 6.7 and Figure 6.8 on a semi-log graph. In each case the unfiltered signal for each time horizon is shown (faded), and the low-pass filtered signal is shown in bold with a low-pass filter gain of 0.1.

The final error for the position and angular components are shown in Table 6.2. Since the back propagated error was squared, the final true error is shown in Table 6.2. While smaller times perform better as expected under the MP assumption, the goal is to find the largest, valid time horizon as it allows for more computational capacity before the predicted pose is reached.

Both Figure 6.10 and Figure 6.11 show the prediction for the time horizon of $1.5s$ for the map

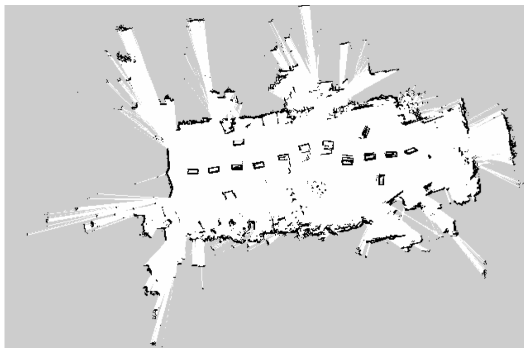
Table 6.2: **Human Intent Training:** Final hold out set errors. Low pass filter gains are 0.1.

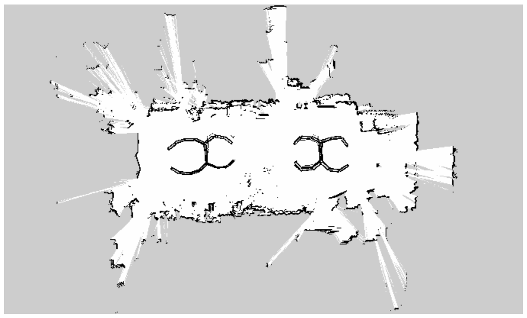
Time horizon (sec)	$\|\boldsymbol{e}_{xy}\|$ (cm) no filter	$\|\boldsymbol{e}_{xy}\|$ (cm) low-pass filtered	$\boldsymbol{e}_{\theta}$ (deg) no filter	$\boldsymbol{e}_{\theta}$ (deg) low-pass filtered
0.5	14.5905	14.3788	4.78913	6.11002
1.0	18.2051	22.2244	9.03606	9.78113
1.5	33.4619	34.829	11.4714	13.9134
2.0	42.4927	47.9171	18.7699	19.1647

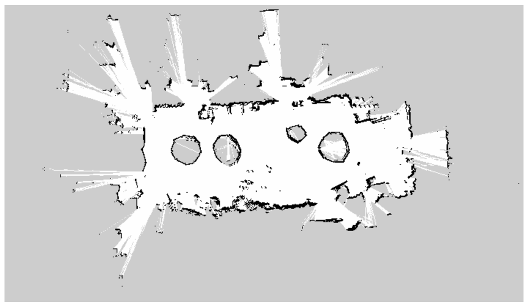
shown in Figure 6.5e. In Figure 6.11 the red arrows indicate the pose of the carried load, and the purple arrows indicate the predicted poses with a time horizon of $1.5s$. While the training error shown in Figure 6.9 represents the average error of randomly selected maps and measured quantities, the representation in Figure 6.11 validates the MP assumption over sequential data. To validate the performance of the neural net prediction, it is compared against integration of the system twist over the corresponding time horizons in Figure 6.12. In Figure 6.12 the MP assumption is used to average the predictions from all maps shown in Figure 6.5. The position and angular error mean and covariance are shown. For position error, the neural network clearly outperforms the velocity integration with increased disparity for longer time horizons. For angular error, the performance is comparable with velocity integration performing slightly better for short time horizons, then for longer time horizons the neural network again outperforms velocity integration.

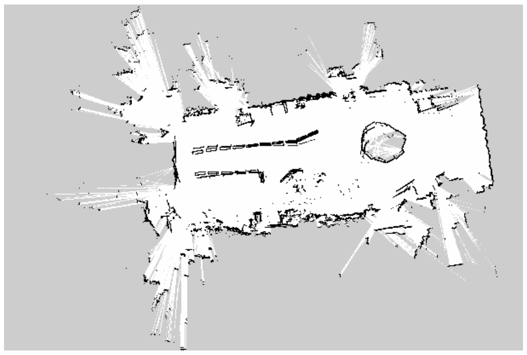
6.5 Conclusion

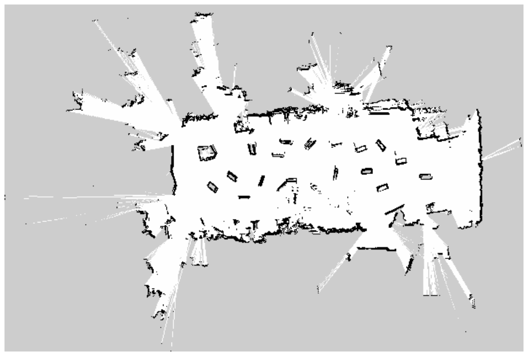
In this chapter a framework is presented and demonstrated to predict the human leader's intended pose on a finite time horizon for the cooperative transport problem. The complete cooperative transport problem between a human leader and robot collaborator is a combination of the methods presented in Chapter 4, Chapter 5 and Chapter 6. Figure 6.13 illustrates this where the human is first observed (Chapter 6), then a collision free trajectory is calculated to the estimated intended pose (Chapter 4), and finally given the grasp map which includes the human and robot, the robot is able to actuate along the specified trajectory while minimizing interaction forces (Chapter 5). During implementation, the robot remains compliant while actuating along the expected path, and continuously updates the expected pose based on new human and environment information.


(a) Dashed lines


(b) Dead ends


(c) Four rings


(d) Hall and diamond


(e) Scattered obstacles

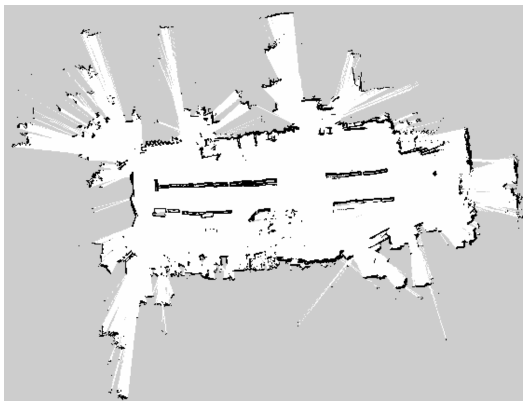

(f) Two hallways

Figure 6.5: Obstacle maps for training human intent inference.

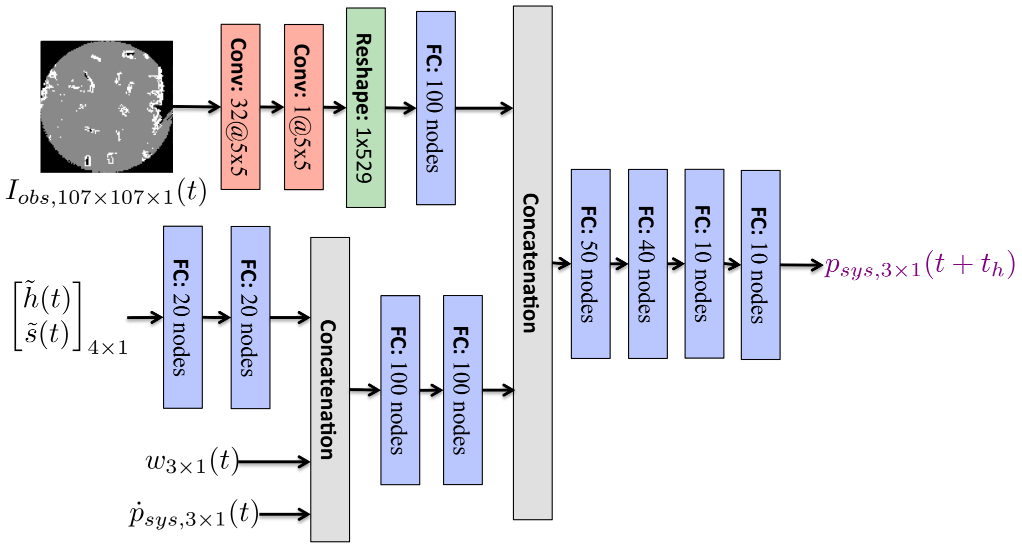


Figure 6.6: Neural net architecture for detecting human intent for cooperative transport. Convolutional layers are denoted with conv. and the number of channels and kernel size are shown with the notation: channel @ kernel size. Fully connected layers are denoted by FC and the number of nodes in the width shown. Leaky ReLU activation functions were used for all nodes. This network was trained by minimizing predicted pose error in $SE(2)$ in units of degrees and centimeters squared. To improve generalization, a dropout rate of 10% was used for convolution and 20% for fully connected layers. The essential intuition for this particular architecture is that the head and step vectors pass through the same layers as they are both related to the expected position. The local map serves as a constraint, with convolutional and fully connected layers working to model the constraint. These are then combined with the wrench which relates to acceleration and system twist for velocity through a set of fully connected layers which models there interdependence for predicting the expected pose on the time horizon.

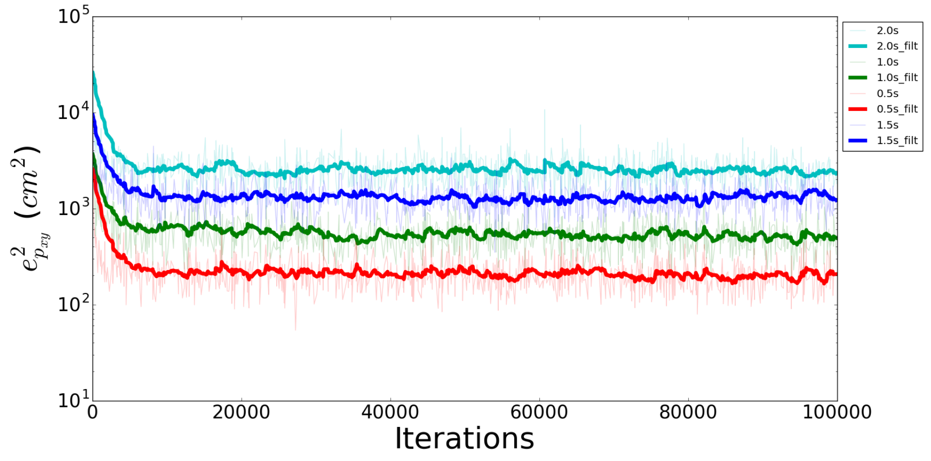


Figure 6.7: NN training error for position error squared, each legend is for horizon time in seconds on hold out set (training data never touched and used to truly evaluate model performance). For times much greater than 2 seconds the Markov process assertion begins to break down as expected in cluttered environments.

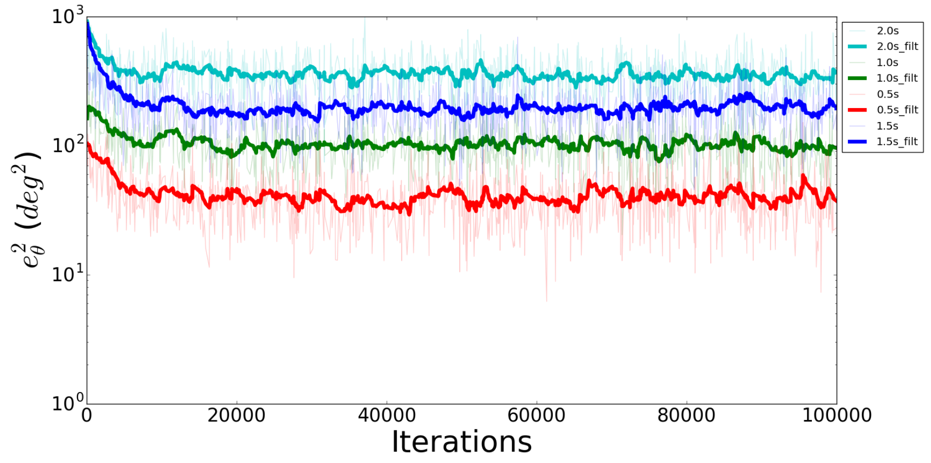


Figure 6.8: NN training error for angular error squared. Each legend is for horizon time in seconds on hold out set.

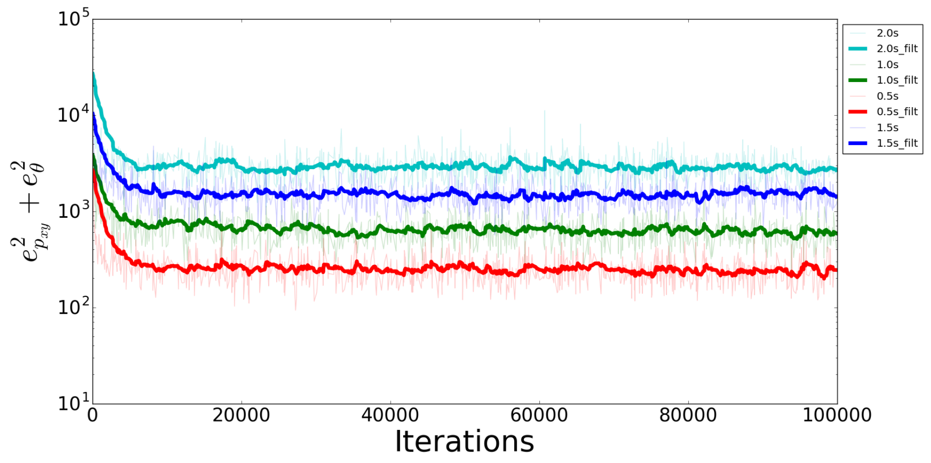


Figure 6.9: NN training error combined for both position and angular error squared, each legend is for horizon time in seconds on hold out set.

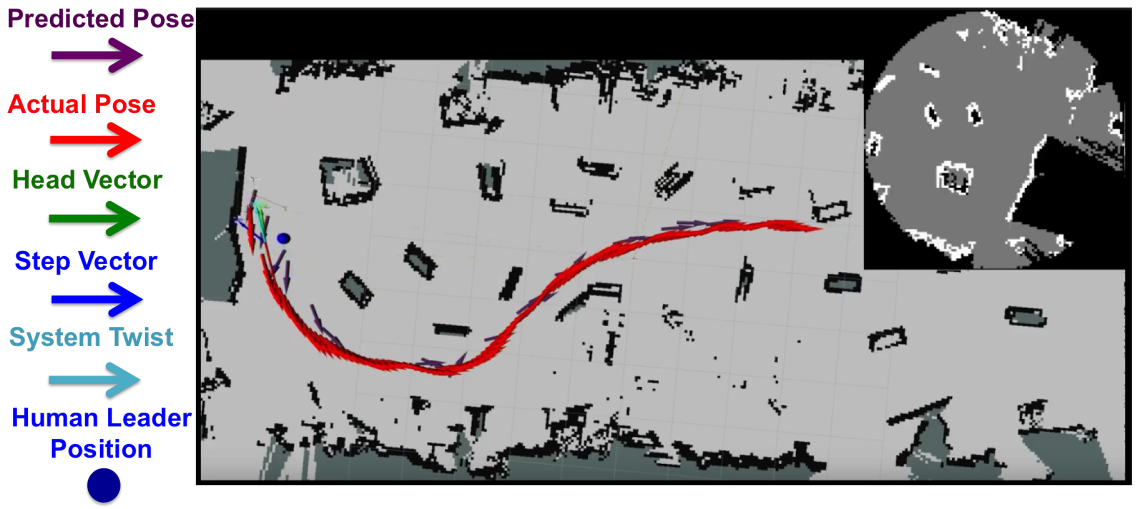


Figure 6.10: Top down view of human intent prediction for a look-ahead time horizon $t_h = 1.5s$. The purple arrows are predicted $SE(2)$ poses, the red arrows are the actual pose of the carried load. This was obtained in Rviz in top-down-view, hence while it may appear that the trajectory goes over obstacles this is an artifact of this view.

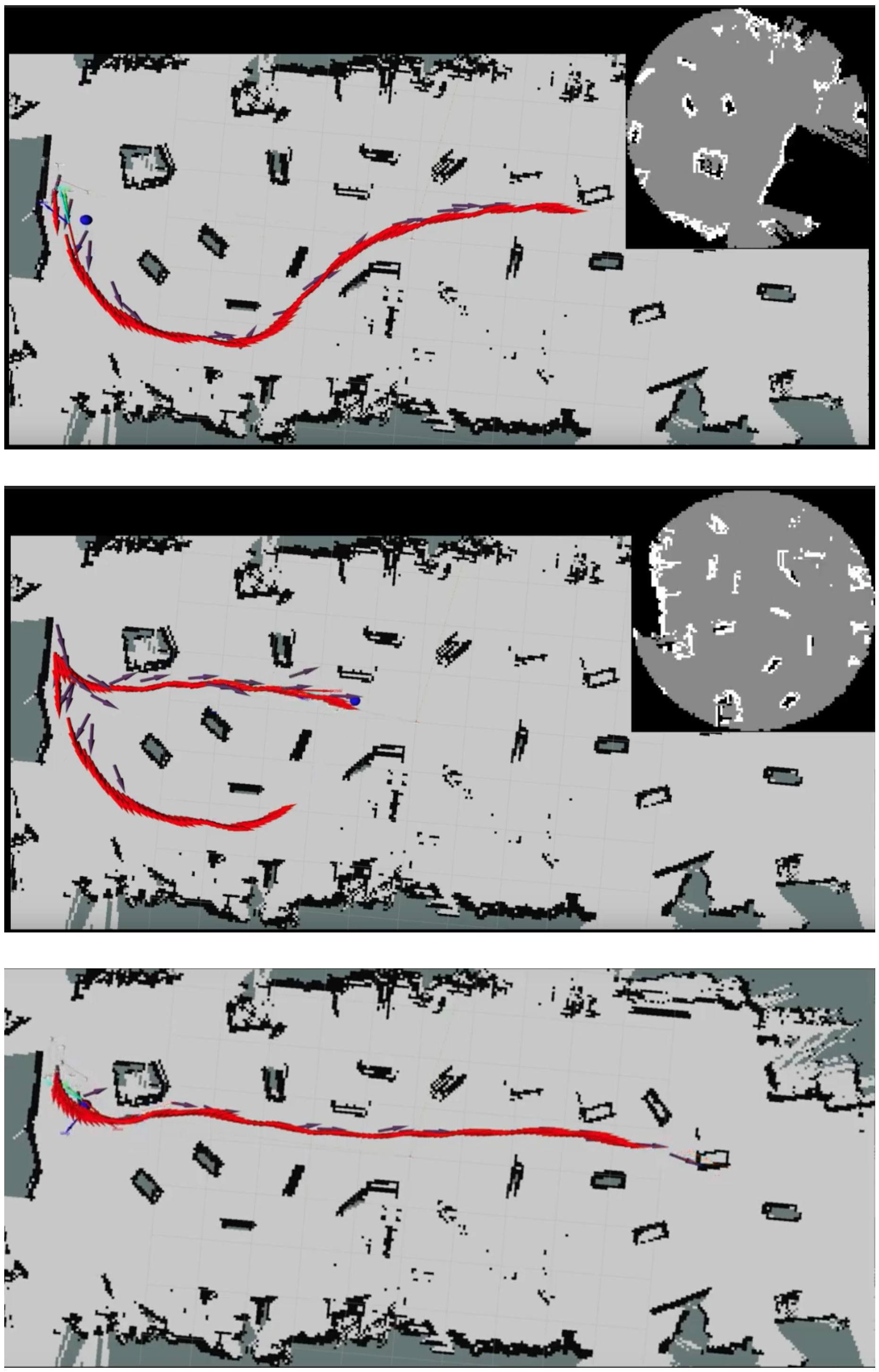
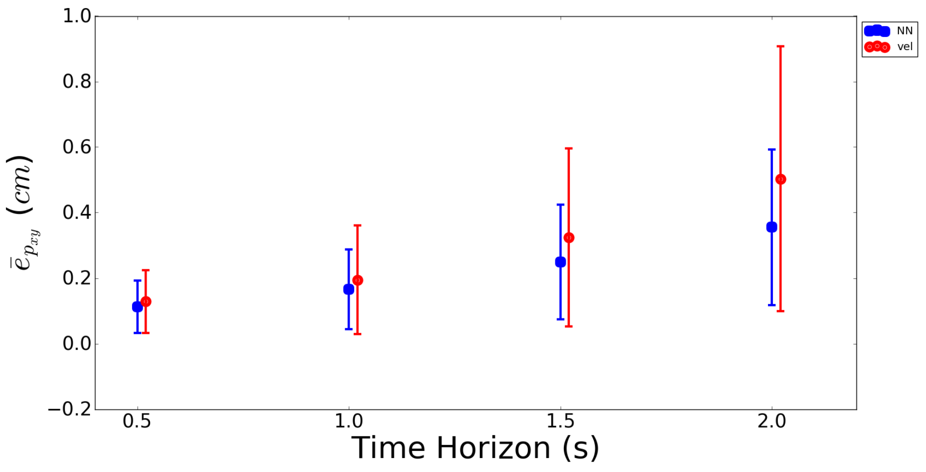


Figure 6.11: Top down view of human intent prediction for a look-ahead time horizon $t_h = 1.5s$. Additional images of the intention predictions performance with same representation as Figure 6.10.



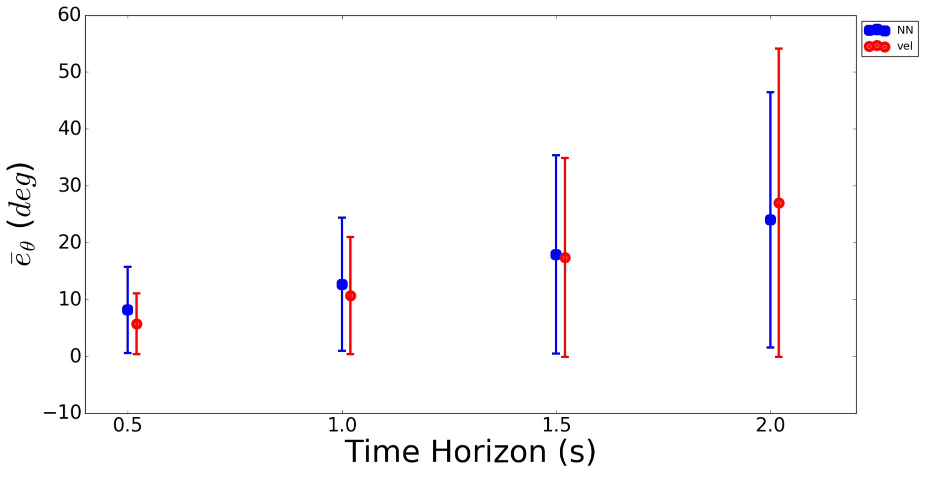


Figure 6.12: Comparison between neural net prediction and velocity integration for expected pose for a given time horizon. Shown are the mean and variance errors for position $\|e_{xy}\|$ and orientation e_θ for all maps shown in Figure 6.5. Neural net outperforms velocity clearly for $\|e_{xy}\|$ for all time horizons, with increasing disparity for longer time horizons. For small time horizons, the performance of the neural net and velocity integration is comparable but for longer horizons the neural net outperforms the velocity integration with smaller average error and tighter variance.

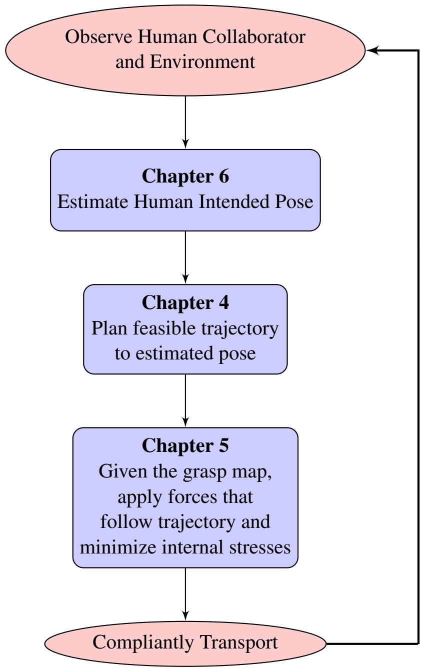

Observe Human Collaborator and Environment
Chapter 6
Estimate Human Intended Pose
Chapter 4
Plan feasible trajectory to estimated pose
Chapter 5
Given the grasp map, apply forces that follow trajectory and minimize internal stresses
Compliantly Transport

Figure 6.13: Human-Robot Cooperative Transport System Algorithm: The human and environment are first observed, then using methods presented in Chapter 6 the human leaders intended pose is estimated. Then using methods presented in Chapter 4, a collision free trajectory is planned to the estimated goal pose. Then the robot actuates along the calculated trajectory using methods presented in Chapter 5. During implementation the robot follower would be compliant while continually observing the human and environment and constantly updating the estimated intended pose.

Part III

Precision Pouring

Chapter 7

Precision Pouring from Known Containers

7.1 Introduction

The chapters in this section consider a robot wet-lab collaborator that assists in rapid experiment preparation for research and development scientists. In this scenario, it is inefficient for scientists to use large batch solution making machines, however considerable time is spent making solutions for experiments. The necessity is for an autonomous robot that is capable of making such small batch solutions while requiring very little environment augmentation as it works alongside the research scientist collaborator. To be effective, the robot must be able to manipulate containers already in use by the wet-lab, as well as pour precisely compared to a human counterpart. During deployment, a robot assistant would approach a workbench to assist the human collaborator. The human collaborator would then indicate the experiment preparation procedure to the robot. The preparation task would require pouring known fluids from an initial open container (pouring container) into a final open container (receiving container). In this work, it is assumed that the fluid is an incompressible, isotropic Newtonian fluid. Even with known fluids, the *inherent challenge is to pour both quickly and precisely within a single attempt.*

In this section, it is always assumed that the receiving container is transparent and the received

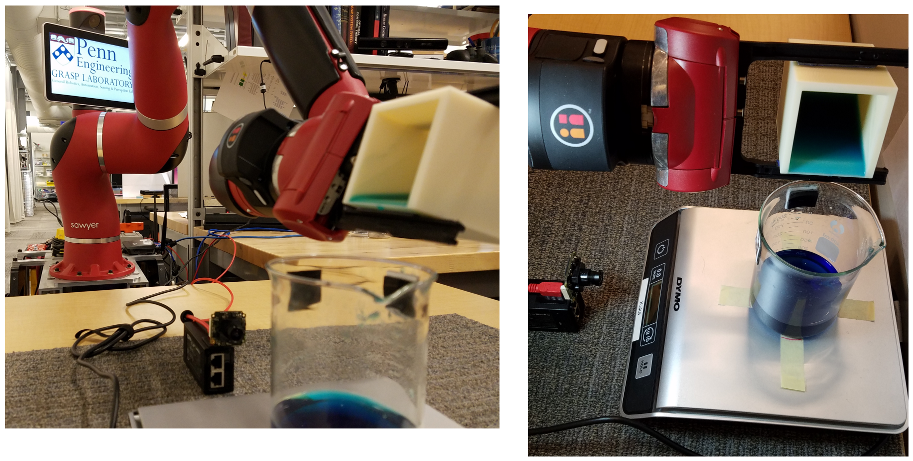

(a) (b)

Figure 7.1: The Rethink Robotics Sawyer manipulator precisely pours colored water into a beaker using visual feedback from a mvBluefox MLC202bc camera.

volume can be measured either visually or using weight. However, two alternative scenarios are considered concerning the pouring container. In Chapter 7 it is assumed that the geometry of the pouring container is known, and an analytical solution is derived to pour quickly and precisely. In Chapter 8 it is asserted that an analytical representation of the pouring container does not exist and the container may be opaque. In this case, the geometry of the pouring container is scanned and used to predict the expected flowrate in order to pour both quickly and precisely in a single attempt.

In this chapter, a simplistic pouring geometry is considered for analytical representation. And a hybrid control strategy is proposed to pour the precise amount of fluid [28].

7.2 General Pouring Problem

Using the Rethink Robotics Sawyer manipulator shown in Figure 7.1a precision pouring is performed using a container of known geometry, colored fluid for easy fluid height detection from a MLC202bc camera and a scale for ground truth as shown in Figure 7.1b. For a pouring container α and receiving container β, the flowrate between the two containers assuming no spillage is

$$q_{fr} = \dot{V}_{\beta}(t + t_d) = -\dot{V}_{\alpha}(t) = -(\dot{V}_{L,\alpha}(t) + \dot{V}_{s,\alpha}(t)). \tag{7.1}$$

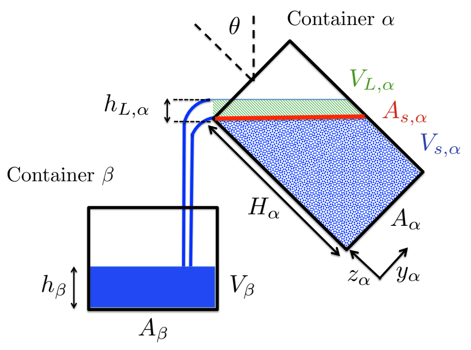


Figure 7.2: The Pouring Problem: For a fluid poured from container α to container β with specified geometric parameters, the goal is to pour a precise amount of fluid using visual or weight feedback based on an analytical model and closed form control.

where t_d is the time delay which for the proposed system is due to the fall-time of the liquid, and $\dot{V}_\beta$ is the volume of the receiving container defined as

$$V_\beta = \int_0^{h_\beta} A_\beta(z_\beta) dz_\beta. \tag{7.2}$$

where h_β is the height of the fluid in the container and $A_\beta(h_\beta)$ is the cross sectional area as a function of height. For the pouring container α, if the container geometry is known, then the volume can be divided into the volume above the pouring lip L

$$V_{L,\alpha} \simeq h_{L,\alpha} A_{s,\alpha}(\theta). \tag{7.3}$$

where $h_{L,\alpha}$ is the height of the fluid above the lip, and $A_{s,\alpha}$ is the horizontal cross-sectional area of the interior of the pouring container that contains the pouring edge as shown in Figure 7.2. And the volume of fluid below the pouring lip is

$$V_{s,\alpha} = \int A_{s,\alpha}(z_\alpha) dz_\alpha. \tag{7.4}$$

assuming that the fall time is negligible, then (7.1) can be expanded to

$$\begin{aligned} q_{fr} &= -\dot{h}_{L,\alpha} A_{s,\alpha} - h_{L,\alpha}\frac{\partial A_{s,\alpha}}{\partial\theta}\dot{\theta} - \frac{\partial V_{s,\alpha}}{\partial\theta}\dot{\theta} \\ &= A_\beta(h_\beta)\dot{h_\beta}. \end{aligned} \tag{7.5}$$

as shown in Figure 7.2. The flowrate over the pouring lip can be found using Bernoulli's principle. The height of the fluid above the pouring lip $h_{L,\alpha}$ is used to define the area of the pouring mouth as $A_L = h_{L,\alpha} L_{L,\alpha}(h_{L,\alpha})$, where $L_{L,\alpha}(h_{L,\alpha})$ is the line of the opening of the mouth at varying heights). For an incompressible, isotropic Newtonian fluid, the height $h_{L,\alpha}$ is related to the flowrate by Bernoulli's principle $\frac{v^2}{2} + gz + \frac{P_{bern}}{\rho} = const.$ where P_{bern}, z, v, ρ are the fluid pressure, height, velocity and density respectively at a particular point in the steady, streamline flow. Considering Bernoulli's principle acting on volume $V_{L,\alpha}$, the fluid at the top surface has no velocity, whereas the fluid at the bottom (height $h_{L,\alpha}$ below the surface) has a velocity $v(h_{L,\alpha}) = \sqrt{2gh_{L,\alpha}}$. As flow rate is defined as $q_{fr}[\frac{m^3}{s}]$ or $[m^2 \cdot \frac{m}{s}]$, then by integrating over the pouring area, the flowrate is

$$q_{fr} = A_L(h_{L,\alpha})v(h_{L,\alpha}) = \int_0^{h_{L,\alpha}} L_{L,\alpha}(h)\sqrt{2gh}dh. \tag{7.6}$$

Differentiating (7.6) with respect to time, then $\dot{h}_{L,\alpha}$ can be expressed in terms of the flowrate and it's derivative $q_{fr}, \dot{q}_{fr}$, which when substituted provides the dynamical equation of the system. The task becomes obtaining the pouring container geometry specific functions for the lip length $L_{L,\alpha}(h_{L,\alpha})$, flow rate $q_{fr}(h_{L,\alpha})$, dividing area $A_{s,\alpha}(\theta)$, and volume below the lip $V_{s,\alpha}(\theta)$.

7.3 Container Geometry Influence on Pouring Dynamics

The main parameters that are considered for the pouring dynamics in (7.5) and (7.6) are the lip length $L_{L,\alpha}(h_{L,\alpha})$, flow rate $q_{fr}(h_{L,\alpha})$, dividing area $A_{s,\alpha}(\theta)$, and volume below the lip $V_{s,\alpha}(\theta)$. For the lip length, consider three cases: a rectangular lip with constant length, a v-shaped lip which has an opening angle φ, and a circular lip shape, where the entire opening has a radius $r_{L,\alpha}$. These

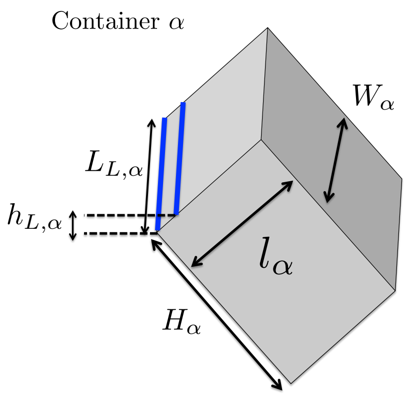


Figure 7.3: Pouring geometry used to derive the analytical model.

lip shape equations become

$$L_{L,\alpha,rect}(h) = L_{L,\alpha} \tag{7.7}$$

$$L_{L,\alpha,vshape}(h) = 2h\cos(\frac{\varphi}{2}) \tag{7.8}$$

$$L_{L,\alpha,circ}(h) = 2\sqrt{h(2r_{L,\alpha} - h)}. \tag{7.9}$$

The flowrate q_{fr} for circular and rectangular lip geometries are

$$q_{fr,rect} = \frac{2}{3}L_{L,\alpha}\sqrt{2g}h_{L,\alpha}^{\frac{3}{2}} \tag{7.10}$$

$$q_{fr,circ} = -\frac{4\sqrt{2g}}{15}\left(128r_{L,\alpha}^5 - 120r_{L,\alpha}^3h_{L,\alpha}^2 + 20r_{L,\alpha}^2h_{L,\alpha}^3 + 30r_{L,\alpha}h_{L,\alpha}^4 - 9h_{L,\alpha}^5\right). \tag{7.11}$$

which when differentiated with respect to time become

$$\dot{q}_{fr,rect} = L_{L,\alpha}\sqrt{2g}h_{L,\alpha}^{\frac{1}{2}}\dot{h}_{L,\alpha} \tag{7.12}$$

$$\dot{q}_{fr,circ} = \frac{-4\sqrt{2g}}{15}\left((-240r_{L,\alpha}^3h_{L,\alpha} + 60r_{L,\alpha}^2h_{L,\alpha}^2 + 120r_{L,\alpha}h_{L,\alpha}^3 - 45h_{L,\alpha}^4)\right)\dot{h}_{L,\alpha}. \tag{7.13}$$

For the dividing area $A_{s,\alpha}$, consider two container geometries: circular and square. In both instances the cross sectional area is constant in body frame z_α. In both cases the dividing area $A_{s,\alpha}$ is defined to consist of a major and minor axis a, b, where rotation occurs about the minor axis b. For a circular container, the area of an ellipse is πab, hence the respective areas are

$$A_{s,\alpha,circ} = \pi a'b = (\pi ab)sec(\theta) \tag{7.14}$$

$$A_{s,\alpha,rect} = a'b = (ab)sec(\theta), \tag{7.15}$$

where a' is the elongated axis as a function of the angle θ. Differentiation with respect to time produces

$$\dot{A}_{s,\alpha,circ} = \pi ab \tan(\theta)\sec(\theta)\dot{\theta} \tag{7.16}$$

$$\dot{A}_{s,\alpha,rect} = ab \tan(\theta)\sec(\theta)\dot{\theta}. \tag{7.17}$$

Note that while other geometries can be found, the volume of fluid below the dividing surface $V_{s,\alpha}$ for rectangular geometry is straight forward. Using the geometry notation shown in Figure 7.3, with container width W_α, length l_α, total height H_α

$$\begin{aligned} V_{s,\alpha,rect} &= \int_0^{l_\alpha}\int_0^{H(y)}\int_0^{W_\alpha} dxdzdy \\ &= \int_0^{l_1} W_\alpha H(y)dy = \int_0^{l_1} W_\alpha(H_\alpha - y\tan(\theta))dy \\ &= W_\alpha H_\alpha l_\alpha - \frac{l_\alpha^2}{2} W_\alpha \tan(\theta). \end{aligned} \tag{7.18}$$

The derivative with respect to time produces

$$\dot{V}_{s,\alpha,rect} = -\frac{l_\alpha^2 W_\alpha}{2}\sec^2(\theta)\dot{\theta}. \tag{7.19}$$

By substituting $h_{L,\alpha}$ from (7.10) into (7.12), then (7.12) can be expressed as

$$\dot{h}_{L,\alpha} = \left(\frac{2}{3}\right)^{\frac{1}{3}} L_{L,\alpha}^{-\frac{2}{3}}(2g)^{-\frac{1}{3}} q_{fr}^{-\frac{1}{3}}\dot{q}_{fr}. \tag{7.20}$$

The container with known geometry used in controls in this experiment is the rectangular geometry with dimensions shown in Figure 7.3. With the rectangular parameterization, the dynamical equation is found by using equations (7.7), (7.10), (7.12), (7.20), (7.15), (7.17), (7.19) and assuming constant A_β to expand (7.5) and solve for $\dot{q}_{fr}$, then substituting the relation

$$\dot{q}_{fr} = A_\beta \ddot{h}_\beta. \tag{7.21}$$

and solving for $\ddot{h}_\beta$ yields

$$\ddot{h}_\beta = E_1(\theta) A_\beta^{\frac{1}{3}} \dot{h_\beta}^{\frac{4}{3}} + \left(E_2(\theta)\dot{h_\beta} + E_3(\theta) A_\beta^{-\frac{2}{3}} \dot{h_\beta}^{\frac{1}{3}} \right) \dot{\theta}. \tag{7.22}$$

$$\begin{aligned}
E_1(\theta) &= -3^{\frac{1}{3}} L_{L,\alpha}^{\frac{2}{3}} g^{\frac{1}{3}} W_\alpha l_\alpha sec(\theta) \\
E_2(\theta) &= -\frac{3}{2}\tan(\theta) \\
E_3(\theta) &= \left(\frac{3}{8}\right)^{\frac{1}{3}} l_\alpha sec(\theta) L_{L,\alpha}^{\frac{2}{3}} g^{\frac{1}{3}}
\end{aligned}$$

as derived in [28].

7.4 Hybrid Control for Precision Pours

The state of the system can be defined as $\begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix}^T = \begin{bmatrix} h_\beta & \dot{h_\beta} & \theta \end{bmatrix}^T$ and input $u = \dot{\theta}$. The dynamics of the system is then represented as

$$\underbrace{\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{bmatrix}}_{\dot{\boldsymbol{x}}} = \underbrace{\begin{bmatrix} x_2 \\ E_1(x_3) A_\beta^{\frac{1}{3}} x_2^{\frac{4}{3}} \\ 0 \end{bmatrix}}_{f(\boldsymbol{x})} + \underbrace{\begin{bmatrix} 0 \\ E_2(x_3)x_2 + E_3(x_3) A_\beta^{-\frac{2}{3}} x_2^{\frac{1}{3}} \\ 1 \end{bmatrix}}_{g(\boldsymbol{x})} u, \tag{7.23}$$

The height is the control objective and is represented in the output equation

$$\boldsymbol{z}(t) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \boldsymbol{x}(t) \tag{7.24}$$

Given a goal final volume of fluid corresponding to a goal height, a minimum jerk trajectory for the fluid described by a 5'th order polynomial can be uniquely defined with known boundary conditions for fluid start and final height, velocity and acceleration. This 5th order polynomial trajectory is described by the function $z_{1,des}(t)$. Defining the error as

$$\boldsymbol{e}_x(t) = \boldsymbol{z}_{1,des}(t) - \boldsymbol{z}_1(t) \tag{7.25}$$

The control problem becomes

$$\begin{aligned} \min_u S &= \int_0^t \boldsymbol{e}_x(s) Q \boldsymbol{e}_x(s) ds \\ \textit{subj to. } \dot{\boldsymbol{x}}(t) &= f(\boldsymbol{x}(t)) + g(\boldsymbol{x}(t))u \end{aligned} \tag{7.26}$$

where Q is a positive semi-definite matrix. The system is feedback linearizable if

$$\begin{bmatrix} g(x) & ad_f g(x) & ad_f^2 g(x) \end{bmatrix} \tag{7.27}$$

is full rank and the span is involutive. Here ad_f is the adjoint of f and is therefore it's lie bracket with g. This is the case when $x_2 \neq 0$ (which means the height must be changing), and the second term in $g(x)$ is only invertible when $x_3 \neq -\frac{\pi}{2}$ or $x_3 \neq \frac{\pi}{2}$. The hybrid control scheme

$$u = \begin{cases} g(x)^{\dagger}(\boldsymbol{s} - f(x)) & x_2 \neq 0 \text{ and } x_3 \in (-\frac{\pi}{2}, \frac{\pi}{2}) \\ sgn(x_3)\delta_{\dot{\theta}} & x_2 = 0 \text{ and } x_3 \in (-\frac{\pi}{2}, \frac{\pi}{2}) \\ 0 & x_3 \notin (-\frac{\pi}{2}, \frac{\pi}{2}), \end{cases} \tag{7.28}$$

where

$$\boldsymbol{s}(t) = \ddot{x}_{1,des}(t) + K_p(x_{1,des}(t) - x_1(t)) + K_d(x_{2,des}(t) - x_2(t)). \tag{7.29}$$

Through feedback linearization, the error dynamics become

$$\dot{\boldsymbol{e}}_x = \begin{bmatrix} 0 & 1 \\ -K_p & -K_d \end{bmatrix} \boldsymbol{e}_x \tag{7.30}$$

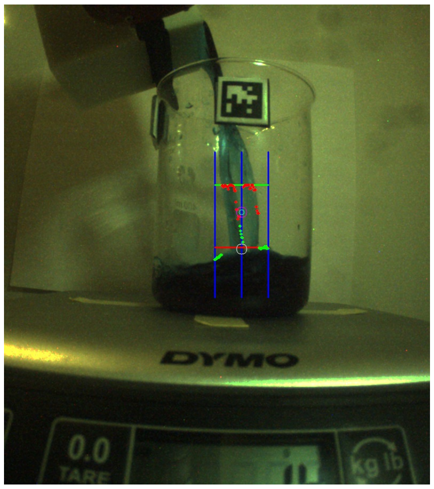
which is asymptotically stable when the following conditions hold: $K_p, K_d > 0$, $K_d^2 > 4K_p$. The error being asymptotically stable about the origin satisfies the optimal control objective in (7.26).

7.5 Results

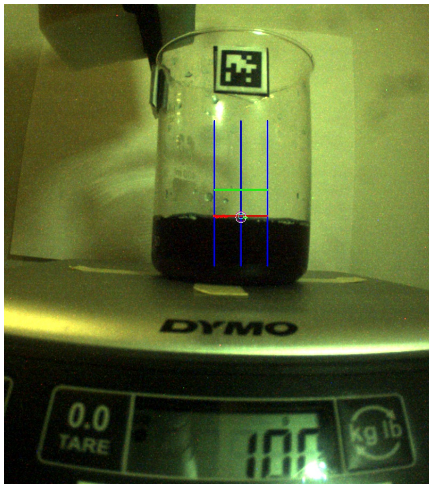
The measured height of the fluid is found using vision to track the fluid. A mvBluefox MLC202bc camera is used and ground truth is established with a DYMO Digital Postal Scale M25. A fiducial is used to locate the beaker in the frame, then an OpenCV online background subtractor which is an implementation of a Gaussian mixture-based background/foreground segmentation algorithm [87] is used. Given the foreground, a Sobel operator is used to find the gradient in the image then K-means clustering is performed on the gradient y-pixels under the fiducial to distinguish between fluid entering the beaker and the rising height of the fluid in the beaker as shown in Figure 7.4a.

In Figure 7.5, volume objectives of 80ml, 100ml, 120ml are poured with 8s, 10s, and 12s trajectories. Note that with longer trajectories and larger volumes the accuracy increases. In Figure 7.6a 50 trials of 100ml pours with 10s trajectories is presented. Contributions to the variance include the fact that the pouring container did not always contain the same initial amount of fluid, hence the hybrid controller provides a constant positive velocity until pouring begins, then the trajectory is initialized at the onset of pouring.

Figure 7.6b shows the error between the visual volume estimation and the reported scale. Across all 50 trials there is a consistent peak in the difference at the beginning of the pours. This is attributed to the fact that because the camera is kept at a constant height, when the fluid level is observed from above, the surface of the fluid can offset the reported height, but as the level of the fluid approaches the height of the camera, the reported value increases in accuracy.

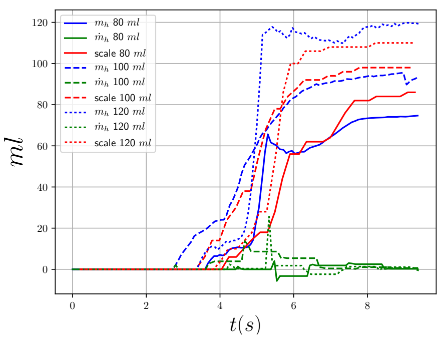


(a)

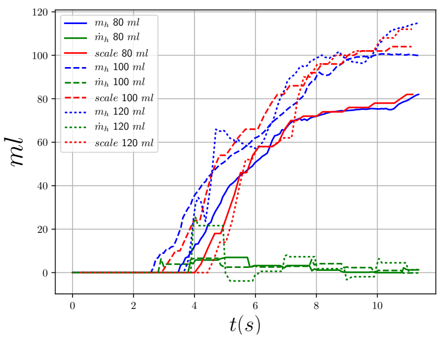


(b)

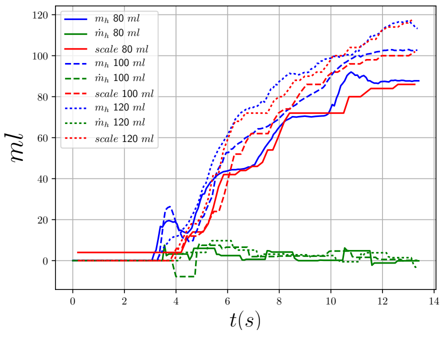
Figure 7.4: Using background subtraction, Sobel gradient detection and K-means clustering (Figure 7.4a), the top of the fluid is tracked for feedback control. In Figure 7.4b the goal was to pour 100ml which was achieved.



(a) 8 second trajectory

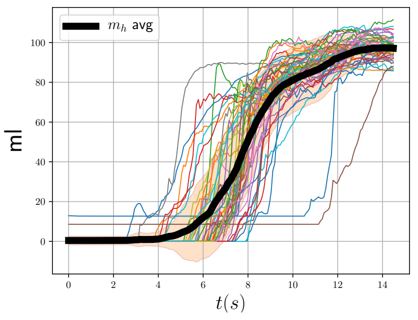


(b) 10 second trajectory

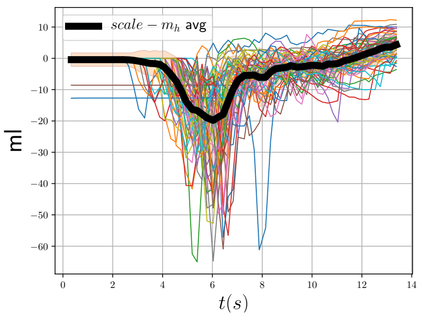


(c) 12 second trajectory

Figure 7.5: Representative experimental trials of pouring. The trials are for pouring 80, 100, and 120ml, over three different time intervals.



(a) 10 second trajectory



(b) 10 second trajectory

Figure 7.6: Figure 7.6a 50 Trials pouring 100ml for 10s trajectories. Black line is average, shaded region is 1st standard deviation. Figure 7.6b average and standard deviation between scale and vision report of height. Consistent dip is due to detection of the top of the fluid at the beginning of the pour from camera perspective registering as higher heights.

7.6 Conclusion

This chapter presents a method of pouring a known incompressible, isotropic Newtonian fluid quickly and precisely from open containers when the geometry of the pouring and receiving containers have known analytical representations. A hybrid controller is presented that is robust to the initial amount of fluid in the pouring container. The controller is also robust to halts in fluid flow which occur due to errors in visual volume estimation caused by sloshing or by the formation of bubbles. In these instances the controller relies on its hybrid nature to re-initiate flow to pour the precise amount of fluid.

When the geometries of all containers are known it is shown that it is possible to derive an analytical model for the flowrate and solve for the controller input that will track the desired trajectory. However, this becomes a challenge when the geometry of the pouring container is unknown and a analytical representation is not provided or easily obtained. Chapter 8 addresses this by first assuming that the pouring container can be scanned. Then by combining online system identification and prior models, an approximate flow rate model is obtained and used for control.

Chapter 8

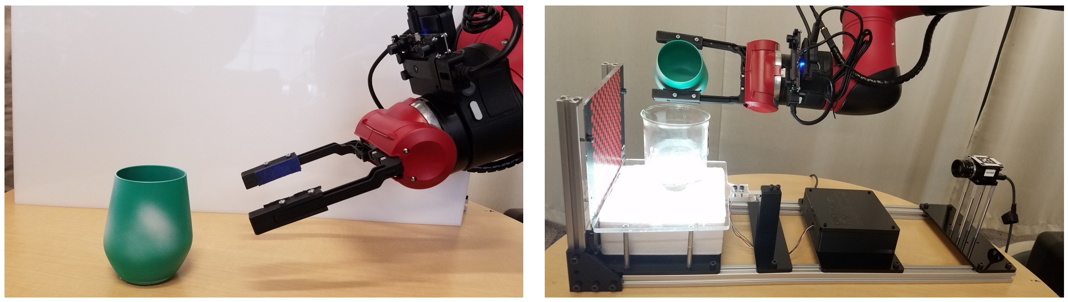
Precision Pouring from Unknown Containers

8.1 Introduction

In this chapter the autonomous precision pouring problem is solved with the relaxed condition that the geometry of the pouring container is unknown. However, in order for this application to be useful, it is necessary that the accuracy of the system not be compromised during system identification. The additional constraint is added that the pours should be relatively quick given an accurately identified model and a known incompressible, isotropic Newtonian fluid.

The precision pouring problem can be decomposed into the observation of the poured fluid, modeling of the flow dynamics and controlling the pouring container to reach a target volume. Previous work has investigated methods of detecting water in flight [80], [61] as well as fluid in a cup when viewed from above classified into 11 fill percentages [45]. The proposed method measures the volume of fluid in the receiver by combining both mass from scale and vision by detecting water pixels in a transparent receiver.

Once the fluid is detected, the flow dynamics between the containers must be modeled. In [72] and [49] system identification on model parameters is performed to improve the performance of derived models. Motion primitives for the pouring task were learned in [34], [69] and [58]. In



(a) (b)

Figure 8.1: The objective is to pour precisely from an unknown container. The pouring rate (which depends on the profile of maximum volume of fluid containable at a given tilt angle) is modeled by estimating the container geometry (Figure 8.1a) and then using model priors and online system identification to identify the profile and pour precisely in a single attempt (Figure 8.1b).

[11] and [38] transformations of points clouds from example containers were morphed to observed containers and a corresponding transformation was applied to the task space trajectory for pouring. In [40] simulated pours are used to learn to mitigate spillage. The proposed approach combines online system identification with model priors leveraging the pouring container geometry, as well as focuses on precise fluid transference assuming no spillage as opposed to just emptying contents.

Once the pouring model has been identified, the system must be controlled to pour the specified volume. In [72], [62], [28] the angular rate of the pouring container is controlled based on the known pouring model. The proposed approach utilizes a hybrid control strategy that incorporates the process model and estimated time delay in the plant. For this application, the proposed approach improves on [62] with an average pour error and time of 38ml and 20 seconds with an accuracy within 10ml and average of 3ml with pour times varying from 20-45 seconds. The proposed approach improves on [72] and [28] in that the robot pours precisely in a single attempt and is not confined to a particular geometry given the pouring container is symmetric.

This work advances the state of the art in autonomous pouring with an effective wet-lab solution preparation system capable of *a*) Leveraging simulated containers and pours to obtain pouring dynamics and expected plant time delay for a new target container. *b*) Pouring target volumes quickly and precisely leveraging system identification and model priors. *c*) Combining vision and mass fluid detection methods to obtain the received volume.

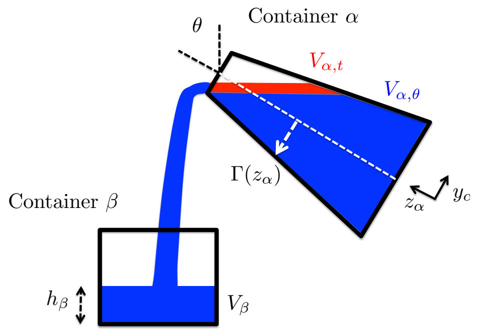


Figure 8.2: The goal is to pour a specified amount of a known fluid from container α to container β quickly and precisely. The receiver volume V_β is sensed (through weight and height h_β) and the geometry of the pourer Γ is estimated. Where Γ describes the radius as a function of height. $V_{\alpha,t}$ and $V_{\alpha,\theta}$ are the transient and steady state volumes in α for a given tilt angle θ.

8.2 Methodology

8.2.1 Problem Formulation

Given a pouring container α, and receiving container β, the goal of this work is to quickly and precisely pour a designated amount of fluid. The general pouring problem is presented in Figure 8.2 where both containers are open, and during pouring only the volume in the receiver V_β is detected. For symmetric containers, Γ specifies the radius of the container as a function of height which is observed before pouring as shown in Figure 8.2. In container α, the volume of fluid above and below the pouring edge are denoted as $V_{\alpha,t}$ and $V_{\alpha,\theta}$ respectively and are the transient and steady state volumes when α is held at tilt angle θ. Let the function $V_{\alpha,\theta}(\theta)$ be denoted as the volume angle profile. The height of fluid in the receiver is denoted as h_β.

The generalized pouring problem is to consider the volumetric flow rate between containers α

and β and the dynamics of the system are

$$\begin{aligned}
\dot{V}_\beta(t+t_{fall}) &= -\dot{V}_\alpha(t) \\
\dot{V}_\beta(t+t_{fall}) &= -\left(\left.\frac{dV_{\alpha,\theta}}{d\theta}\right|_{\theta(t)} + \left.\frac{dV_{\alpha,t}}{d\theta}\right|_{\theta(t)}\right)\frac{d\theta}{dt} \\
\dot{V}_\beta(t+t_d) &= -\underbrace{\left.\frac{dV_{\alpha,\theta}}{d\theta}\right|_{\theta(t)}}_{V^{(1)}_{\alpha,\theta}}\frac{d\theta}{dt},
\end{aligned} \tag{8.1}$$

where t_{fall} is the fall time of the fluid from α to β, t_d is the time delay for both the fall time and dissipation of $V_{\alpha,t}$, θ and $\frac{d\theta}{dt}$ are the angular position and velocity of the pouring container. $V^{(1)}_{\alpha,\theta}$ denotes the derivative of $V_{\alpha,\theta}$ with respect to θ. Since the receiving container is only observed, the volume angle profile must be expressed in terms of container β. At steady state the following holds

$$V^{(1)}_{\alpha,\theta} = -V^{(1)}_{\beta,\theta}, \tag{8.2}$$

and for bounded velocity pours this equivalence is asserted in (8.1). Defining the volumetric error as

$$e_V(t) = V_{\beta,des}(t) - V_\beta(t), \tag{8.3}$$

the goal is to determine the control input $u(t) = \frac{d\theta}{dt}$ that achieves the desired volume

$$\min_{u(t)} \int_0^t e_V(s)^2 ds. \tag{8.4}$$

The desired trajectory $V_{\beta,des}(t)$ is specified using a trapezoidal trajectory generation algorithm to minimize pouring time, subject to velocity and acceleration constraints and is discussed further in Section 8.2.4. The generalized time delay is an unknown function of the pouring container geometry, the tilt angle and control input

$$t_d(t) = f_t\left(\Gamma, \theta(t), u(t-t_u)\right). \tag{8.5}$$

Define t_u to be the period of the controller, as the current time delay is approximated with the last commanded velocity. The time delay approximates the dissipation of $V_{\alpha,t}$ (which is a function of Γ, θ and u) and t_{fall}. Making the following assumptions *a*) That the pours are slow enough that substitution of (8.2) into (8.1) is valid. *b*) There is no spillage during pouring and sloshing is insignificant for a sigmoid desired trajectory and the specified maximum angular rotation rate. *c*) The viscosity of the fluid is known, and at steady state the fluid conforms to the geometry of the container. *d*) The geometry of the receiving container is known. *e*) There is enough fluid in the pouring container required to reach the specified target volume in the receiver. *f*) The fall time can be approximated by a small constant. *g*) That similarity in container geometry Γ correlates to similarity in volume profile $V_{\alpha,\theta}$. *h*) The pouring container has a symmetric edge profile for simplistic implementation in container scanning. But the method extends to any container geometry.

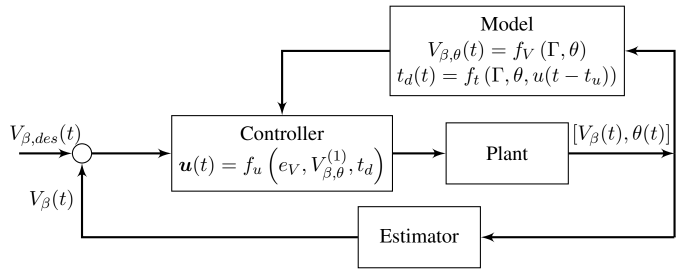

Model
$V_{\beta,\theta}(t) = f_V(\Gamma,\theta)$
$t_d(t) = f_t(\Gamma,\theta,u(t-t_u))$
$V_{\beta,des}(t)$
Controller
$\boldsymbol{u}(t) = f_u\left(e_V, V^{(1)}_{\beta,\theta}, t_d\right)$
Plant
$[V_\beta(t),\theta(t)]$
$V_\beta(t)$
Estimator

Figure 8.3: Pouring Control Diagram. The desired volume is specified by trajectory generator, the controller f_u processes this and the model approximations (f_V and f_t which must be determined) in order to control the angular rate of the pouring container.

The system control diagram is shown in Figure 8.3. The models for $V_{\beta,\theta}$ and t_d are informed by scanning the pouring container geometry $\Gamma(z_\alpha)$ as in Figure 8.2.

8.2.2 Model Learning

Define the error (or distance) between two symmetric container geometries Γ_1, Γ_2 as

$$e_\Gamma = \int_0^1 \left(\Gamma_1(H_{\Gamma,1}s) - \Gamma_2(H_{\Gamma,2}s)\right)^2 ds \tag{8.6}$$

where $H_{\Gamma,1}$ and $H_{\Gamma,2}$ are the respective heights of the containers and with the radial function $\Gamma(z_\alpha)$, the function $\Gamma(H_\Gamma s)$ defines the radius for $s \in [0,1]$ for $z_\alpha \in [0,H]$. The assumption states that

$$\lim_{e_\Gamma \to 0} V_{\beta,\theta,1} = V_{\beta,\theta,2}, \tag{8.7}$$

and is extended to the time delay in (8.5) for a given θ, u. The training set of pouring container geometries are artificially generated and used to simulate pouring with known fluid properties as shown in Figure 8.5. The geometry of the pouring container is scanned before pouring, and the top ten nearest neighbors are then selected based on minimal distance e_Γ. This is then used to generate a mixture probability $p_{stat,j}$ by

$$p_{stat,j} = \frac{e_{\Gamma,j}^{-1}}{\sum_i e_{\Gamma,i}^{-1}}. \tag{8.8}$$

If the new container profile can be interpolated from the training set, then this mixture model will effectively describe the new container profile with adequate resolution in the space of example containers. However in practice this is a strong assertion which is relaxed through adjusting the kernel variance in addition to the mixture probability in semi-parametric model approximation.

Combined Parametric and Non-Parametric Approximations

The maximum volume profile $V_{\beta,\theta}$ is modeled using three methods: parametric, non-parametric and semi-parametric. The parametric method approximates $V_{\beta,\theta}$ using a polynomial of 9th degree

$$V_{\beta,\theta}(\theta) = f_V(\theta) = \sum_{i=0}^{N} \boldsymbol{c}_i \theta^i \tag{8.9}$$

whose coefficients $\boldsymbol{c}$ minimize the following functional

$$
\begin{aligned}
S_\theta = & \underbrace{\sum_{j=1}^{C}\left(V_{\beta,\theta,j} - \sum_{i=0}^{N} \boldsymbol{c}_i \theta_j^i\right)^2}_{\text{Residual}} + \underbrace{k_1 \sum_{i=1}^{N} \boldsymbol{c}_i^2}_{\text{Regulator}} \\
& + \underbrace{k_2 \exp\left(-\sum_{m=1}^{M}\sum_{i=1}^{N} i\boldsymbol{c}_i \theta_m^{i-1}\right)}_{\text{Soft Constraint}}.
\end{aligned} \tag{8.10}
$$

The residual term fits the polynomial to C volume observations in the receiver by minimizing the squared error, the regulator term ensures the N coefficients do not diverge with positive gain k_1, and the soft constraint ensures the polynomial derivative is positive at M control points (locations enforcing constraint) evenly spaced over the entire pouring angle domain which respects the physics that volume in the receiver β is strictly increasing with positive gain k_2.

The non-parametric method uses a Gaussian process (GP) with the radial basis kernel function with white noise:

$$
\kappa(\theta_i, \theta_j) = \sigma_\kappa^2 \exp\left(-\frac{(\theta_i - \theta_j)^2}{l_\kappa^2}\right) + \eta. \tag{8.11}
$$

The variance is related to the edge profile error through $\sigma_{\kappa,j} = k_3 e_{\Gamma,j}^{-1}$ with positive gain k_3, and l_κ is a distance scale factor. The white noise η in the kernel is associated with the volume milliliter measurement error. The covariance matrices are defined as

$$
K_{i,j}^* = \kappa(\theta_{test,i}, \theta_{train,j}) \tag{8.12}
$$

$$
K_{i,j} = \kappa(\theta_{train,i}, \theta_{train,j}) \tag{8.13}
$$

and the information matrix as

$$
L_{GP} = (K + \delta^2 I)^{-1}. \tag{8.14}
$$

With these terms, define the non-parametric estimate

$$
V_{\beta,\theta}(\theta) = f_V(\Gamma, \theta) = \sum_j p_{stat,j} K_j^* L_{GP,j} \boldsymbol{V_{\beta,\theta,train,j}}. \tag{8.15}
$$

The limitation of the parametric method is that it does not leverage prior knowledge of similar containers when available. Likewise, the limitation of the non-parametric method is that it cannot adapt when the new container is drastically different from the training set. By combining these methods, their positive attributes can be leveraged for better performance across a larger range of containers. This is done by making the parametric profile estimation the mean of the GP:

$$V_{\beta,\theta}(\theta) = \sum_{i=0}^{N} \boldsymbol{c}_i \theta^i + \sum_{j=1}^{10} p_{stat,j} K_j^* L_{GP,j} \left(\boldsymbol{V_{\beta,\theta,train,j}} - \sum_{i=0}^{N} \boldsymbol{c}_i \boldsymbol{\theta^i_{train,j}} \right). \tag{8.16}$$

The additional component is the last term which uses the current parametric model to evaluate the training pours. The error of the parametric function and true training volume is used to adjust the expected value for $V_{\beta,\theta}$. If the parametric function is a very good approximation, the error between $\boldsymbol{V_{\beta,\theta,train}}$ and the function evaluation becomes zero and the parametric mean dominates. If the parametric mean is a poor fit, but the container is interpolated well between example containers, then the GP terms accommodates this error with sufficient sampling of $\boldsymbol{\theta_{train}}$.

8.2.3 Real-Time Volume Estimation

A combination of visual feedback and weight measurement is used to track the volume of the fluid in the receiving container. For visual volume detection, the receiver is first located the container in the scene via a fiducial. Once the container is localized, a neural network is used to find the probability that each pixel is water. As in [28], clustering is used to distinguish between fluid entering the receiver and contained rising fluid. With the known cross section this provides an estimate of the volume of water in the receiving container to be considered with the measured mass. A load cell is used to obtain the weight of fluid in the receiver and the volume estimate from both vision and weight are combined using a Kalman filter whose details are in Section 8.3.1.

8.2.4 Trajectory Planning and Control

Trajectory Generation

Trajectories for volume in the receiving container $V_{\beta,des}(t)$ are determined using the user specified target volume, and specified upper and lower maximum velocities dependent on the current residual of the $V_{\beta,\theta}$ approximation. The area under the trapezoid is the target volume. Respecting maximum allowable accelerations, time optimal trajectories are computed in a similar implementation to [55]. A key difference is the calculation of the maximum velocity is a sigmoid function of the mean residual:

$$\bar{r}_{res} = \frac{1}{C}\sum_{j}^{C} \left| V_{\beta,\theta,pred,j} - V_{\beta,\theta,meas,j} \right|, \tag{8.17}$$

where this is evaluated for every new accumulated measurement set C, and new model $V_{\beta,\theta}$ which adjusts $V_{\beta,\theta,pred,j}$. Given this residual, the maximum velocity is calculated using the following function

$$\dot{V}_{\beta,max}(\bar{r}_{res}) = \dot{V}_{\beta,ml} + \frac{\dot{V}_{\beta,mu}}{1 + \frac{\dot{V}_{\beta,ml}}{(\dot{V}_{\beta,mu} - \dot{V}_{\beta,ml})}\left(\exp\left(k_4 \frac{\bar{r}_{res}}{r_{res,max}}\right)\right)}, \tag{8.18}$$

where $\dot{V}_{\beta,ml}, \dot{V}_{\beta,mu}$ are the lower and upper bounds on allowable maximum velocities and $\bar{r}_{res} \in [0,\infty)$ is the residual for fitting $V_{\beta,\theta}$. The term $r_{res,max}$ is a threshold residual ensuring for large residual that $\dot{V}_{\beta,max} \simeq \dot{V}_{\beta,ml}$.

Proposed Controller

Given the trajectory generator specifies $V_{\beta,des}(t)$, the controller then uses the volume error e_V along with model estimates of the volume profile and time delay $V_{\beta,\theta}$, t_d, to calculate the control output which is the angular velocity of the container.

The hybrid controller is dependent on the following conditions (a): $\theta \in [0,\pi]$, (b): $V^{(1)}_{\beta,\theta}(\theta) > 0$, (c): $\dot{V}_\beta > 0$, and (d) $e_V > 0$

$$u(t) = \begin{cases} \left(V^{(1)}_{\beta,\theta}(\theta(t))\right)^{-1}(K_p e_V(t+t_d)) & \text{if: } (a) \wedge (b) \wedge (c) \\ \delta_\omega & \text{if: } (a) \wedge (d) \wedge \neg((b) \wedge (c)) \\ 0 & \text{if: } \neg(a). \end{cases} \tag{8.19}$$

The third state stops motion if the angle is outside the acceptable regions of operation. The second state (re)initiates the pour. Hence when the container is in the operation domain from condition (a) the first state is obtained.

Controller Stability

A stability analysis is presented for the first hybrid state, as the second state always results in the first state unless there is not enough fluid in the container to pour the target volume which violates a base assumption. Given the current time t, the future error at $t + t_d$ for time delay t_d is

$$\begin{aligned} e_V(t+t_d) &= V_{\beta,des}(t+t_d) - V_\beta(t+t_d) \\ &= V_{\beta,des}(t+t_d) - \int_0^t V^{(1)}_{\beta,\theta}(\theta(s))u(s)ds. \end{aligned} \tag{8.20}$$

Consider the Lyapunov function $\mathcal{V} = \frac{1}{2}e_V^2$, the system is asymptotically stable if $\dot{\mathcal{V}} < 0$:

$$\begin{aligned} \dot{\mathcal{V}} &= e_V \dot{e_V} \\ &= e_V\left(\dot{V}_{\beta,des}(t+t_d) - V^{(1)}_{\beta,\theta}(\theta(t))u(t)\right). \end{aligned} \tag{8.21}$$

Let

$$u(t) = \left(V^{(1)}_{\beta,\theta}(\theta(t))\right)^{-1} K_p e_V(t+t_d) \tag{8.22}$$

then if

$$\dot{V}_{\beta,des}(t+t_d) = 0, \tag{8.23}$$

then (8.21) reduces to

$$-K_p e_V^2 < 0, \tag{8.24}$$

which is true if $K_p > 0$ and the system is asymptotically stable. If $\dot{V}_{\beta,des}(t+t_d) > 0$ then (8.21) stability condition becomes

$$e_V(t+t_d) > K_p^{-1}\dot{V}_{\beta,des}(t+t_d). \tag{8.25}$$

Hence the system will trail until the condition of (8.25) is true, then when $\dot{V}_{\beta,des} = 0$ the error will attenuate to zero. Note that a larger K_p will reduce the magnitude of e_V required for asymptotic stability in (8.25).

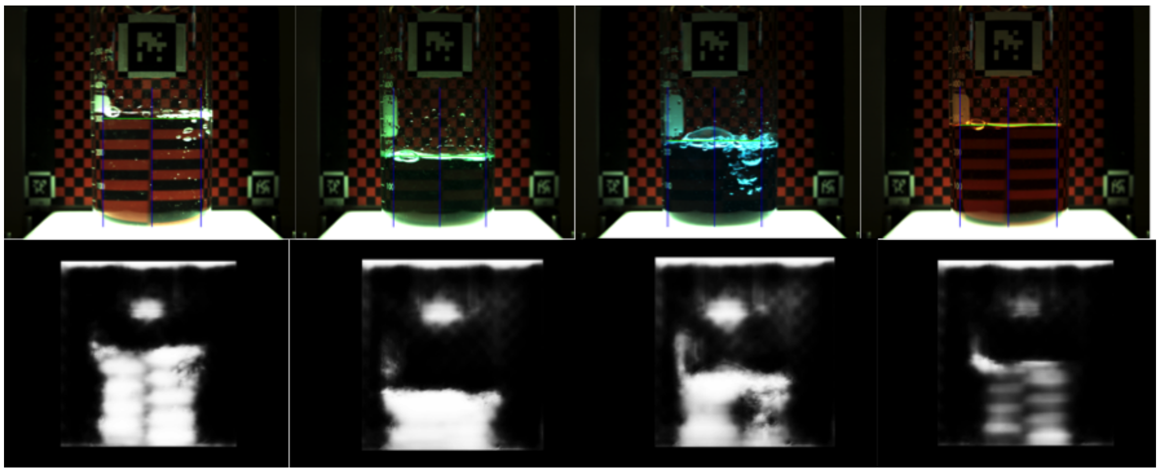


Figure 8.4: Volume in the receiver is detected combining vision and weight. The volume is visually estimated by using a fiducial to locate the receiver of known geometry then a network detects fluid pixels. The detection is robust to fluid color and transparency.

8.3 Implementation

8.3.1 Volume Measurement

Volume Detection

The receiving container is placed on an illuminated stand. A 1280×1040 pixel Point Grey RGB camera is mounted horizontally facing the container, and a checkerboard background shown in Figure 8.1b is used to leverage distortion and occlusion for fluid detection. The network architecture from Holistically-Nested Edge Detection is used, it is a network that extracts multi-scale features from VGGNet and uses them for pixel-wise edge detection [79]. The trained network detects pixel masks that show the locations of water (instead of detecting edges as in [79]) and runs at 21Hz on a cropped 390×412 pixel image based on the fiducial location.The visual system is able to calculate the height of the water h_β for many different colors of water, ranging from clear to completely opaque as shown in Figure 8.4.

The beaker is placed $42 \pm 5cm$ from the camera, the camera is $6cm$ above the platform, and the receiver diameter is known (in this experiment 3.64cm) as shown in (Figure 8.1b). This makes the top of the fluid visible if the volume of fluid is below 250ml, with a maximal error of 1.04cm translating to 43ml at the onset of pouring. To mitigate this effect, similar triangles are used to

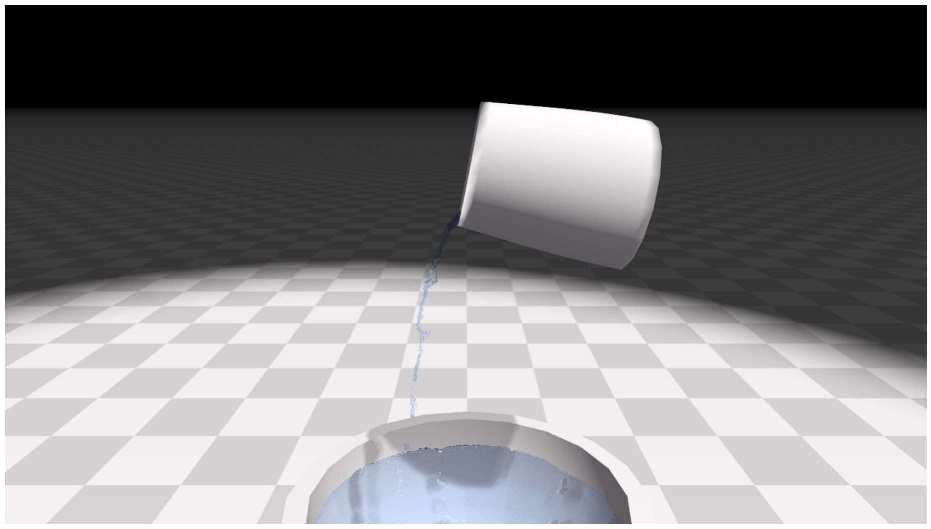


Figure 8.5: Artificial containers are generated to provide a simulation of pouring liquid of known properties in NVIDIA FleX [41].

determine the true height of the container as the fiducial provides both the beakers distance from the camera, and pixel-metric scale factor in the fiducial plane. This geometric adjustment is utilized when the height of fluid is below the center pixel of the camera (which for this beaker correlates to below 250ml). A Uxcell 5kg load cell with an Arduino Uno micro-controller is used to detect the weight of the fluid in the receiving container which runs at 12Hz. The receiver container sits on a suspended platform shown in Figure 8.1b which is a cantilever with the load cell.

Sensor Fusion

Both the volume estimate from vision and scale are combined using a Kalman filter running at 30Hz to approximate the volume in the receiving container. The associated uncertainties for the scale and vision are 1.5ml and 15ml respectively, the process noise was set to be 0.02ml. For the vision, this is due to an 20 pixel variance in detection of fluid height.

8.3.2 Simulated Pouring

To obtain simulated pours, the physics engine NVIDIA FleX is used to match the fluid properties of water for different containers. NVIDIA Flex is a particle based unified graphics solver from [41], used to simulate pouring liquids from different containers. A total of 1792 symmetric containers were randomly generated and scaled between 5 and 20cm. Each container is poured once in simulation (Figure 8.5) and the $V_{\beta,\theta}$ profile is generated by filling the container to the brim with the liquid particles and slowly rotating the container. At each time step, the quantity of liquid inside the container is measured by counting the number of particles that were inside the container's mesh. Parameters in the FleX software were empirically chosen to match behavior for real container geometries for water at room temperature.

8.3.3 Volume Estimation and Attenuation

Establishing a reference frame at the bottom, center of the pouring container which is assumed to be a surface of revolution (SOR). Given the 128 points along the edge of the container, the edge-profile is defined by the set of vectors consisting of each point height and radius. This edge-profile is then compared with the edge profiles of simulated containers, and the closest top 10 containers are selected. For trajectory generation, the parameter was set as $k_4 = 8$ in (8.18).

8.3.4 Volume Profile and Time Delay Estimation

Container Edge Extraction

The geometry of the pouring container is obtained to compare to simulated container geometries to approximate the volume profile. For symmetric containers it is sufficient to scan half of the container to extrapolate the entire geometry. The PMD Technologies Pico Flexx time-of-flight depth sensor is used to extract the point cloud of the container. RANSAC is used extract the table points and approximate the container with a set of cylinders every 1.5cm. The edge profile is then extracted using a Gaussian process. This is shown in Figure 8.6, where Figure 8.6a shows the container, Figure 8.6b shows the extracted points and the fitted surface, and Figure 8.6c shows the fitted cylinders radii and height and the resulting 128 points from the GP fitting. During data collection, the surface of the containers were augmented with form fitting paper for better performance of the time of flight sensor.

Time Delay

The time delay is modeled in simulation by rotating fully filled containers at randomly chosen, constant angular velocities until they reached randomly chosen stop angles. The time delay was defined as time to reach 20% of the initial $V_{\alpha,t}$ from the stopping point. A total of 3,888 trials were simulated for the time delay. A neural network consisting of three convolutional layers for the edge profile and then five fully connected layers combining the convolutional output and the containers height, tilt angle and angular velocity is used to predict the time delay by approximating f_t in (8.5).

8.3.5 Pouring System

The full state machine shown in Figure 8.7, was implemented on the KUKA LBR iiwa manipulator for the KUKA innovation award and can be found on YouTube: “Finalist Spotlight - Precise Robotic Dispenser System - KUKA Innovation Award 2018", and “Kuka Innovation Award finalist: Precise Robotic Dispenser System". In these experiments, for brevity, the container edge-profile extraction and precision pouring were the focus. A video demonstrating this method can be found on at “Autonomous Precision Pouring from Unknown Containers”

8.4 Results and Discussion

8.4.1 Volume Profile Estimation Method Comparison

Three representative containers are shown in Figure 8.8, with the container classifications along with their final volume error performance for each maximum volume profile estimation method. The statistical data is represented using violin plots showing the data distribution along with box plots consisting of data minimum and maximum as thin black line, first and third quartile as solid black line, and the median as white dot. Figure 8.8a shows container 1 which is a small flask, in Figure 8.8b container 2 is a small cylinder, in Figure 8.8c container 3 is a tall flask. For container 1, Figure 8.8a shows the training container along with their mixture probabilities. The second row of Figure 8.8 shows the final volume error for each method while pouring 100ml of fluid. For container 1 and container 2 in Figures 8.8d and 8.8e each method statistic was generated using 30 trials for each method, statistics for container 3 where generated from 30 trials of param, and 10 trials for semi and non-param. For 200ml pours each method for all three containers were generated from 10 trials each. In Figure 8.8 the general trend is that for 100ml pours the parameterized method usually performed better than semi and non-parameterized methods, with these two methods being comparable in performance. However, for the larger volume 200ml pour the semi and non-parameterized methods were more consistent compared to the 100ml pour and performed better than the parameterized pour. All of the containers consistently started with 250ml of fluid before each pour. Another notable aspect is that better performing methods usually had a larger pour time than the lower performing counter parts for the same container and target volume. A negative example is shown in Figure 8.9 where the priors are used from container 1 in Figure 8.8a. Here Figure 8.9b shows each method performance with 30 trials each, and Figure 8.9c shows each method with 10 trials each. Note that while the trend for 100ml is similar to that in Figure 8.8, the spread of the semi and non-parametric methods is much greater. This is an important example as it demonstrates the capacity of the proposed system when a container is far from the training set of containers. In both Figures 8.9b and 8.9c, the contribution of the parametric method influences the semi-parametric performance making it comparable to the parametric performance. The parametric method performance varies with the container as it is performing online system identification, this is evident in how parametric performed worse for larger volumes for containers 1 and 2 however improved for container 3. If valid model priors are known then the performance across container volumes can be more consistent as seen in the non-parametric methods with accurate priors. The semi-parametric method usually interpolates these performances with sufficient overlap with the highest performing method.

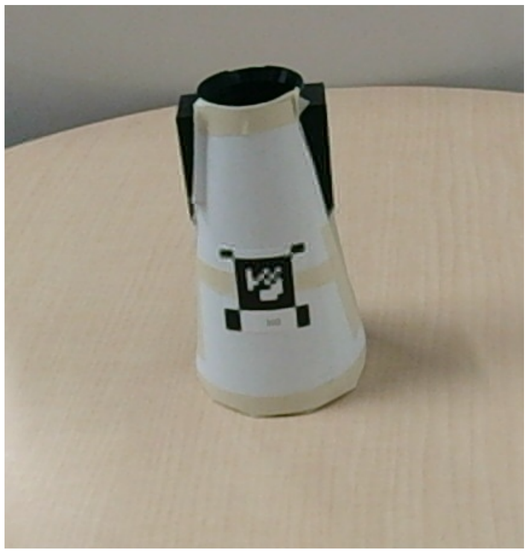
8.4.2 Semi-Parametric Estimation for Diverse Containers

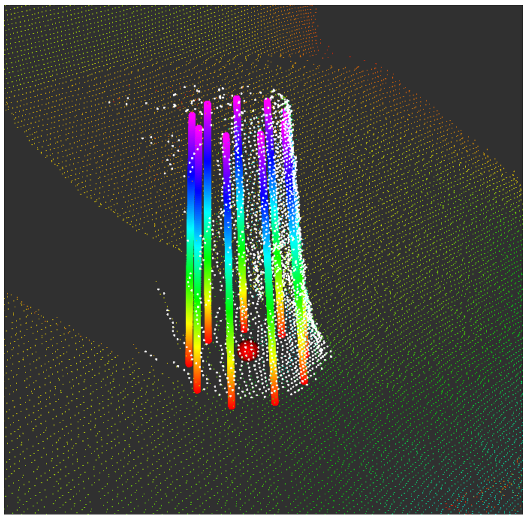
It is shown that the semi-parametric combines the qualities of the other methods by leveraging online system identification and model priors. Due to the strong priors in Figure 8.8a the semi-parametric method prediction of $V_{\beta,\theta}$ was very close in prediction to the non-parametric method for container 1. These profiles vary more for the parametric method and semi-parametric method when the container is sufficiently far from the training set. The semi-parametric method was used on a total of 13 containers pouring target volumes of 100ml with 10 trials each. The final volumetric error for each container is shown versus each containers height in Figure 8.10. The mean and variance is shown for each container and a Gaussian process is fit with a radial basis function kernel to demonstrate the increase in error vs height with associated variance. This trend is attributed to the fact that the lower maximum velocity used in the trapezoidal trajectory generator was constant for all containers. This means that the slowest angular rate was constant for the containers which reduces the control authority for larger containers. This can be remedied by making the lower maximum velocity a function of container height.

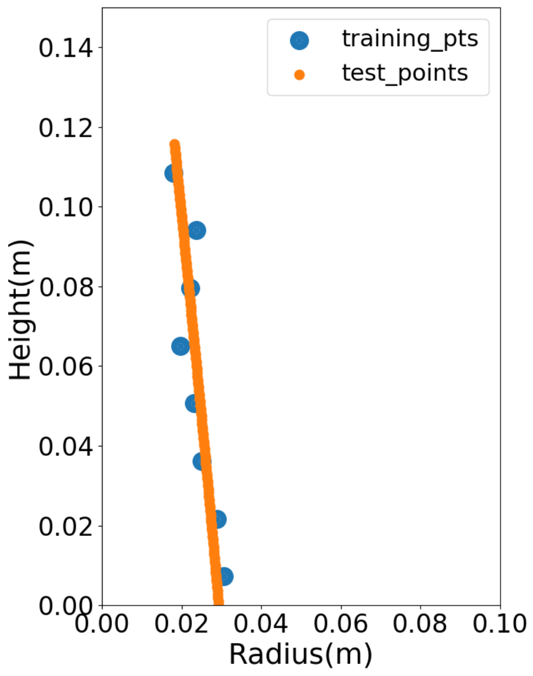
In Figure 8.11 the relationship between final error and final pour time is shown for the same pours shown in Figure 8.10. Over the majority of pours, the performance is largely under 5ml error and between 20 to 45 seconds for 100ml pours. The main two outliers to are those with largest height as shown in Figure 8.10. Therefore with the remedy of making the lower maximum velocity a function of container height, larger containers would also be expected to perform within 5ml error. However this remedy while decreasing final error would also increase the pour time, therefore requiring task based optimization to determine appropriate parameters for best performance. In practice, the volume profile for containers observed before can be stored and utilized as an additional prior therefore increasing the performance with continued operation.

8.5 Conclusions

A system capable of pouring fluids from new containers accurately and quickly in a single attempt is presented. This is done by comparing the profile of the new container to simulated containers pouring an incompressible, isotropic Newtonian fluid with known properties. By defining the maximum volume profile as the maximum fluid containable at a given angle, it is asserted that containers with similar geometries have similar maximum volume profiles. This closeness in geometry is used to combine priors in a mixture model to estimate the maximum volume profile required for control. The fluid is detected in the receiving container using both weight and visual detection of the fluid. A hybrid controller is proposed that accounts for time delay in the plant and attenuates volumetric error given a specified volume trajectory from an optimal time, trapezoidal trajectory generator that accounts for the residual in the volume profile estimation. It is shown that by combining online system identification and model priors through a Gaussian process, the performance is maximized with the specified system without tuning parameters for a given container. It is shown that for constant minimum, maximum desired volume velocity the performance of under 5ml error and between 20 to 45 second pours for the majority of containers can be achieved. For larger containers the accuracy can be increased while sacrificing pour time. This system is demonstrated on the Rethink Robotics Sawyer manipulator as well as an implementation on the KUKA LBR iiwa manipulator. Future directions may include increasing the complexity of receiving glass detection as well as relaxing the stated assumptions by mitigating spillage, and expanding this implementation to non-symmetric containers.


(a)


(b)



(c)

Figure 8.6: Container geometry is scanned, where it is assume that it is a surface of revolution and the geometry can be represented with the edge profile Figure 8.6c. A time of flight sensor is used to extract the edge-profile. For simplicity, the time of flight sensor is used, an alternative method for extracting edge profiles from surfaces of revolution is discussed in [54].

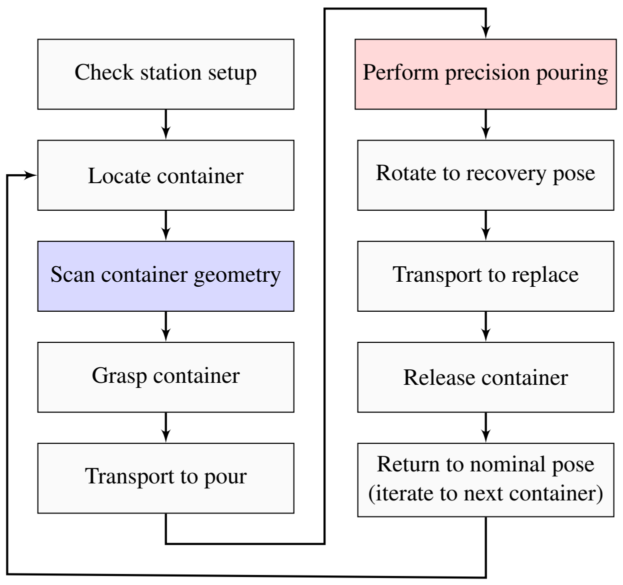
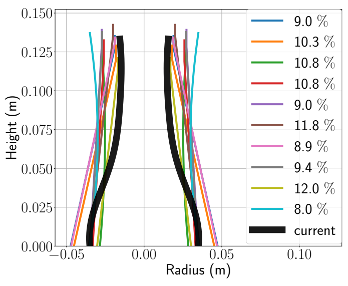
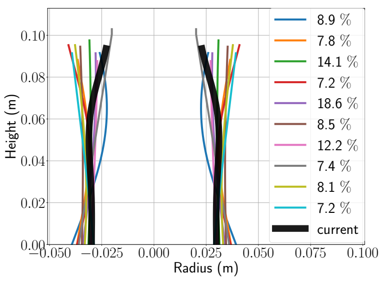


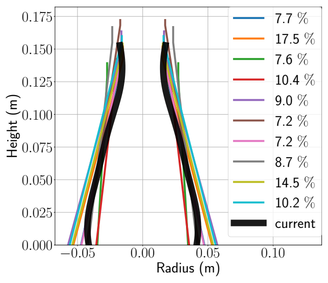
Figure 8.7: System pipeline for autonomous pouring



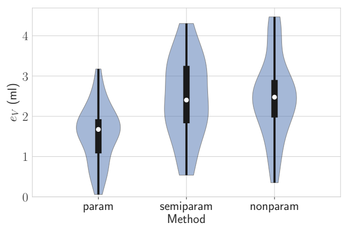
(a)



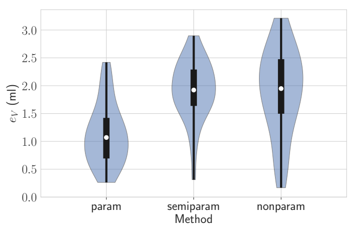
(b)



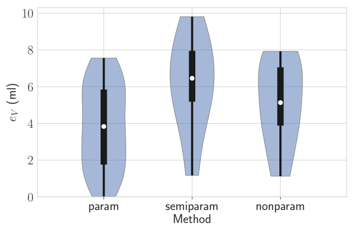
(c)



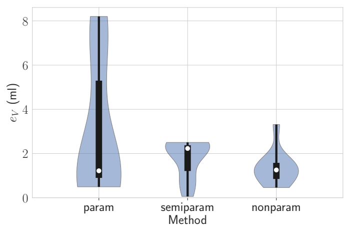
(d)



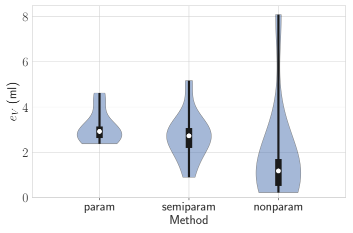
(e)



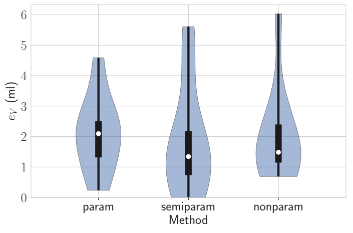
(f)



(g)

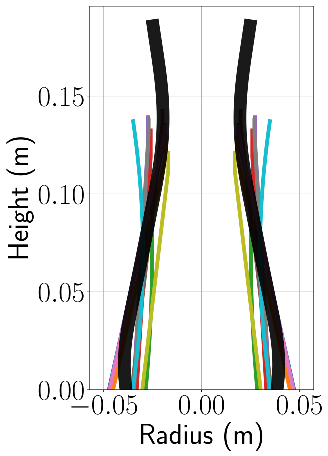


(h)

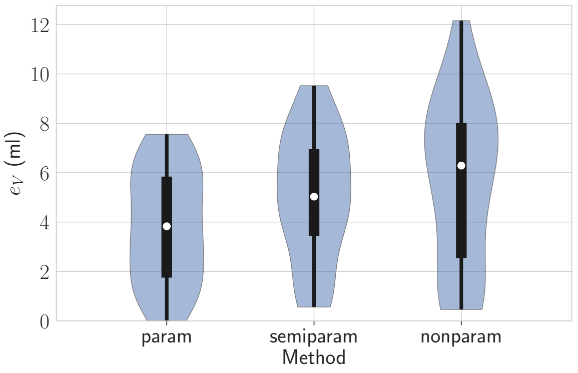


(i)

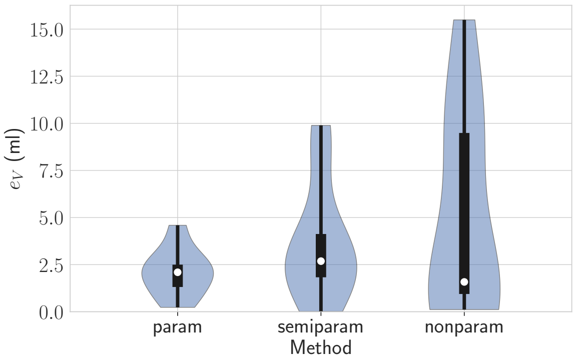
Figure 8.8: Container profiles and pour statistics: containers 1, 2 and 3 are 8.8a, 8.8b and 8.8c respectively, with their statistics directly below each container. Percentages are the mixture probabilities defined in (8.8). Final volume error statistics are shown for a 100ml target volume in the second row for each method (30 pours each), and 200ml target volume in the third row (10 pours each). Each method statistics are shown using violin plots for distribution and inset box plot with mean and quartiles. A total of 360 pours are shown.



(a)



(b)



(c)

Figure 8.9: Container 3 negative example with weak priors along with method statistical data for pours of 100ml in 8.9b and 200ml in 8.9c.

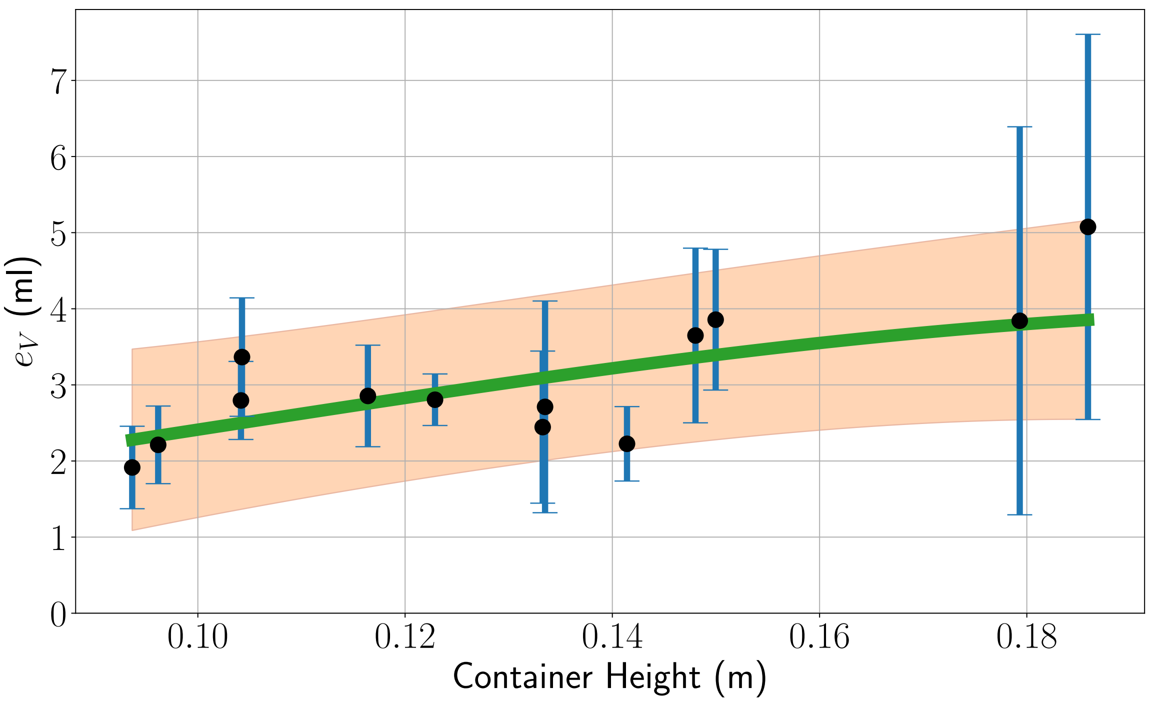


Figure 8.10: Semi-parametric method: final error versus container height for 13 containers. Mean and variance are shown for 10 pours of 100ml each. Trend line and variance fit shows the general increase in error with container height attributed to constant lower bound angular velocity. A Gaussian process is used to generate this fit with the kernel defined in (3.116) with $\sigma_\kappa = 5.7$ and $l_\kappa = 0.12$.

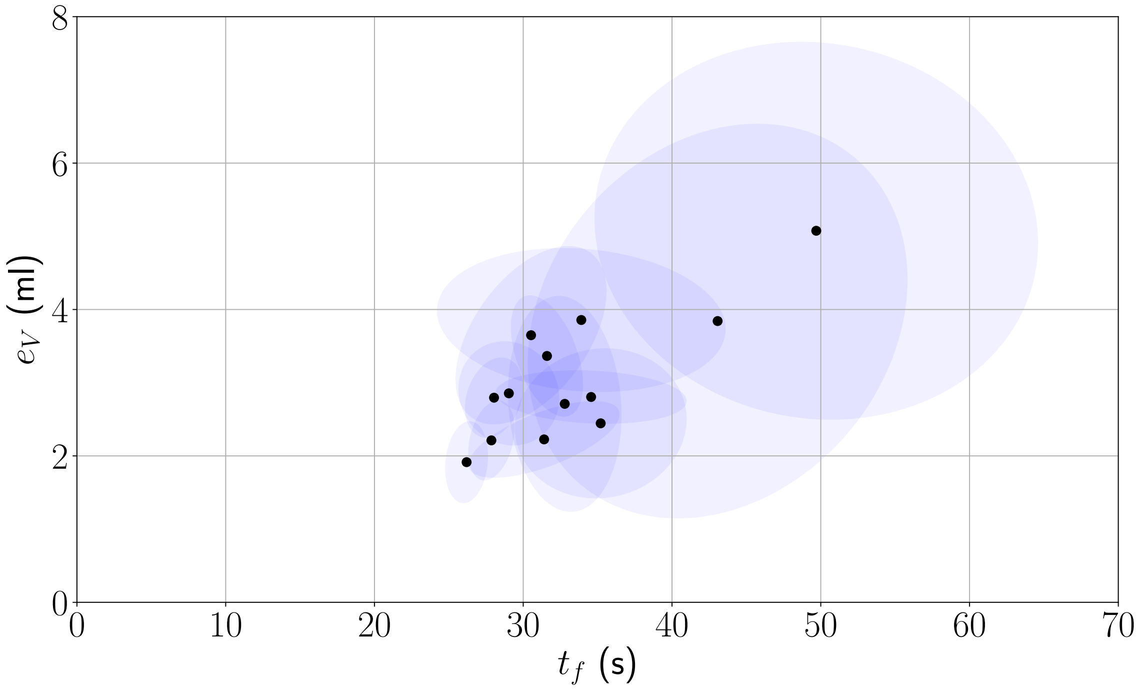


Figure 8.11: Semi-parametric method: final error versus pour time for 13 containers pouring 100ml with 10 trials each. Mean and covariance are shown. Larger error corresponds to larger container height in Figure 8.10.

Part IV

Conclusions

Chapter 9

Contributions and Future Work

9.1 Contributions

As robotics transitions to applications in which human augmentation is required, an emerging area of research is making robots effective collaborators and assistants. In order to become effective assistants, robot must have the capacity to complete complex tasks in unstructured environments. This work specifically focuses on tasks relevant to a wet-lab robotic assistant, but the underlying theory extends to many robotic assistant applications. The challenges addressed in this work are the modeling of complex tasks and designing control strategies based on the estimated model. The two main tasks considered are cooperative transport and precision pouring for rapid experiment preparation.

The first task is cooperative transport with both robotic and human teammates. The problem is challenging when multiple robots and humans must transport an object efficiently in a cluttered environment with limitations on communication. Here efficient transport is described by minimizing energy through traversing the optimal path or minimizing internal stresses in the carried object. The cooperative transport problem for a robot collaborator can be decomposed into: detecting the goal pose for the robot, planning a collision free trajectory to the goal and executing the trajectory such that internal stresses are minimized in the carried object. Goal selection is most interesting when there is a human leader, but they are unable to directly convey the goal location to a robot follower. A contribution of this work is a proposed neural net architecture that is informed by the underlying physics of sensed quantities to predict the human leaders intended pose. Because the architecture is informed by the underlying physics minimal data compared to similar work is required to obtain a well performing model. For planning a feasible trajectory to the identified goal, a contribution of this work is a free-space decomposition algorithm that is computationally efficient as it is design specifically for determining path optimality. Once the trajectory has been determined, the robot must actuate with teammates along the trajectory in an energy efficient manner. This is achieved by minimizing interaction forces (internal stresses) in the carried load that do not contribute to motion. If the grasp map (location of grasps) of all teammates is known to the robot, then the necessary action can be calculated given the described controller. A contribution of this work is in the case that the grasp map is not known by every robot but robots are able to communicate a small amount of information. By reaching consensus with other robot teammates on a small solution vector, robots are able to minimize interaction forces and carry the object efficiently. All of these contributions allow for a framework for a robotic assistant capable of transporting a load with a human counterpart in a cluttered environment while predicting the humans intended goal pose on a small time horizon and using this prediction to minimize internal stresses on the carried load and conserve energy. An underlying strength and consistency in these approaches is that in each case an underlying physics principle has been exploited to make the model more effective and efficient.

The second task is precision pouring. It is a challenging problem because the robot assistant must pour a known incompressible, isotropic Newtonian fluid quickly and precisely from open containers. When the geometries of the pouring and receiving containers are known with an analytical representation, the pouring dynamics must be developed and the appropriate controller must be designed in order to pour precisely. When the geometry of the pouring container is unknown, then the model of the pouring dynamics must be identified in addition to utilizing the appropriate controller. For the case when both geometries have analytical representation, a contribution of this work is the development of the physics model and the hybrid controller that allows for quick, precise pours. For the case where the pouring container geometry is unknown, the contribution of this work is to present a method to extrapolate a physical model from a single scan and perform online system identification in addition to leveraging limited model priors from simulated containers to estimate the underlying physical model. The additional contribution is the design of a hybrid control strategy that leverages this estimated model to pour quickly and precisely in a single attempt. All of these contributions allow for a framework that allows the human collaborator to describe the pouring experiment to the robot and the robot executes, pouring both quickly and precisely. The entire framework pipeline was demonstrated at the KUKA innovation award competition at the Hannover Messe in April 2018. Again, the concept that is the strength and is consistent for all contributions is that in each case key underlying physical principle(s) have been exploited to make the model more effective and efficient by combining model-based and data-driven approaches for best performance.

9.2 Future Work

There are many directions for future work for robotic assistants and specifically for a wet-lab assistant. In the area of precision pouring next steps are relaxing task assumptions and requiring the modeling of more variables that increase the capacity of a deployed system performing this task. An example of this is considering motions that mitigate spillage during pouring regardless of the pouring container edge configuration. Another consideration is exploring the modeling dimension of different fluids. In this work the fluid properties were known to be that of water at room temperature, but for a long-term deployed system it is useful for the robot to be effective with a range of fluid properties. Another direction to explore is the manipulation of powders, specifically scooping or pouring in order to add additives to experiment fluid solutions.

In the area of cooperative transport, one immediate extension is considering a model that considers the temporal effect during transport which would allow for predictions on longer time horizons. One way to do this would be to use a recurrent neural network as a 'history' of previously measured quantities would influence the current prediction. Another extension is making the human intent estimation model robust to various carrying configurations, example being what if the human carried with their back to the robot, determine if the robot can still make effective predictions. In addition, a further consideration is that of re-grasping an object with the human leader in order to reach a target grasp map configuration required to traverse the environment or place the object in a final goal pose.

The underlying theme that connects each of the contributions in this work is leveraging compact, physics based models to complete complex tasks. A natural extension is consideration for how a robot can autonomously deduce these simplistic but accurate dynamical models of the complex task in order to correctly plan and control to achieve the task objectives in the unstructured environment.

9.3 Concluding Remarks

This dissertation presents methods of executing complex tasks in unstructured environments for robot collaborators in the context of tasks relevant for a wet-lab assistant. Task models are presented that lend themselves to adaptation via data driven-models with priors based on knowledge of the modeled process. The tasks considered are cooperative transport and precision pouring. These are the beginning of a suite of capabilities that will be required for wet-lab robotic assistants, and the utilized approaches for modeling complex tasks and methods of control are applicable and useful for the design and deployment of collaborative robots for a broad class of tasks.

The methods developed here can be generalized for new tasks. It is assumed that the task process model is never fully known, but some inherent structure in the task can be exploited. For a process model manifold that is continuous with low curvature, system identification can leverage the inherent structure and be coupled with model priors from data utilized in a framework such as Bayesian modeling approach (in this work a Gaussian process). Because the model prior data is maintained, a distance function can be designed to assess the quality of prediction. This can be used to allow the model to quickly adapt in the presence of good model priors and use system identification more when these priors are not sufficient. This approach was exemplified in Chapter 8. When the process model manifold is continuous but has high curvature, statistical modeling approaches like the Gaussian process become expensive in areas of high curvature as the generating data must be maintained. Here it is advantageous to utilize a neural network with adequate architecture. This is because the computational graph is capable of representing the highly curvature manifold of the process manifold with the static computational graph structure, allowing for an efficient model representation. Often with high curvature process model manifolds it is harder to exploit knowledge of the inherent structure. However this knowledge can usually be used as a guide when designing the neural net architecture. As the generating training data is not maintained in deployment but instead encoded into the computational graph, it is essential that the neural network be trained on the coverage of the state space expected to be visited during operation. Therefore knowledge of the process models inherent structure can guide the engineer on state space regions to focus on when generating training data to ensure the resultant neural network adequately describes the useful state space. This was exemplified in Chapter 6.

Exciting next steps are addressing similarly complex tasks in the case of a wet lab assistant. Designing a fundamental framework that would allow a robot to reason about the task and the environment, leveraging classical understanding of kinematics and dynamics necessary for control. Allowing the robot to become more effective as a human collaborator capable of anticipating the humans desires in a wide range of task and scenarios, which would both enable the human to complete a desired task and amplifying their abilities.

Bibliography

[1] D. J. Agravante, A. Cherubini, A. Bussy, P. Gergondet, and A. Kheddar, "Collaborative human-humanoid carrying using vision and haptic sensing," in *Proceedings - IEEE International Conference on Robotics and Automation*. IEEE, may 2014, pp. 607–612. 10, 11

[2] A. Alsaibie and W. Singhose, "Experimental testing of liquid slosh suppression in a suspended container with compound-pendulum dynamics," *2013 9th Asian Control Conference, ASCC 2013*, pp. 2–7, jun 2013. 13

[3] H. Arai, T. Takubo, Y. Hayashibara, and K. Tanie, "Human-robot cooperative manipulation using a virtual nonholonomic constraint," *Proceedings-IEEE International Conference on Robotics and Automation*, vol. 4, no. April, pp. 4063–4069, 2000. 68

[4] W. Aribowo, T. Yamashita, and K. Terashima, "Integrated trajectory planning and sloshing suppression for three-dimensional motion of liquid container transfer robot arm," *Journal of Robotics*, vol. 2015, p. 3, 2015. 13

[5] T. Baldacchino, E. J. Cross, K. Worden, and J. Rowson, "Variational Bayesian mixture of experts models and sensitivity analysis for nonlinear dynamical systems," *Mechanical Systems and Signal Processing*, vol. 66-67, pp. 178–200, jan 2016. 46

[6] J. Baraglia, M. Cakmak, Y. Nagai, R. P. Rao, and M. Asada, "Efficient human-robot collaboration: when should a robot take initiative?" *The International Journal of Robotics Research*, vol. 36, no. 7, pp. 563–579, 2017. 12

[7] A. Bauer, D. Wollherr, and M. Buss, "Human - Robot Collaboration : A Survey," *International Journal of Humanoid Robotics*, vol. 05, no. 01, pp. 47–66, 2007. 9, 11

[8] M. Beetz, U. Klank, I. Kresse, A. Maldonado, L. Mösenlechner, D. Pangercic, T. Ruhr, and M. Tenorth, "Robotic roommates making pancakes," in *IEEE-RAS International Conference on Humanoid Robots*, oct 2011, pp. 529–536. 9, 12

[9] R. Bellman, "The theory of dynamic programming," *Bulletin of the American Mathematical Society*, vol. 60, no. 6, pp. 503–515, 1954. 33

[10] S. P. Bhattacharyya, H. Chapellat, and L. H. Keel, "Robust control: the parametric approach," in *Printice Hall information and system science series*. Elsevier, 1995. 38

[11] S. Brandi, O. Kroemer, and J. Peters, "Generalizing pouring actions between objects using warped parameters," in *IEEE-RAS International Conference on Humanoid Robots*, vol. 2015-Febru, nov 2015, pp. 616–621. 13, 99

[12] M. Cakmak and A. L. Thomaz, “Designing robot learners that ask good questions,” in *Proceedings of the seventh annual ACM/IEEE international conference on Human-Robot Interaction - HRI ’12*, ser. HRI ’12. New York, NY, USA: ACM, 2012, p. 17. 10

[13] B. Christopher, *Pattern Recognition and Machine Learning*. Springer-Verlag New York, 2006. 40

[14] J. H. Chung, S. A. Velinsky, and R. A. Hess, “Interaction Control of a Redundant Mobile Manipulator,” *The International Journal of Robotics Research*, vol. 17, no. 12, pp. 1302–1309, 1998. 12

[15] R. Deits and R. Tedrake, “Computing large convex regions of obstacle-free space through semidefinite programming,” *Springer Tracts in Advanced Robotics*, vol. 107, pp. 109–124, 2015. x, 52, 55, 56, 58

[16] ——, “Efficient mixed-integer planning for UAVs in cluttered environments,” *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2015-June, no. June, pp. 42–49, 2015. 12

[17] J. Desai and V. Kumar, “Nonholonomic motion planning for multiple mobile manipulators,” in *Proceedings of International Conference on Robotics and Automation*, vol. 4, apr 1997, pp. 3409–3414. 12

[18] P. Dorato, C. T. Abdallah, V. Cerone, and D. H. Jacobson, *Linear-quadratic control: an introduction*. Prentice Hall Englewood Cliffs, NJ, 1995. 36

[19] Duy Nguyen-Tuong and J. Peters, “Using model knowledge for learning inverse dynamics,” in *2010 IEEE International Conference on Robotics and Automation*, may 2010, pp. 2677–2682. 46

[20] G. Fanelli, T. Weise, J. Gall, and L. Van Gool, “Real Time Head Pose Estimation from Consumer Depth Cameras,” in *Pattern Recognition*, R. Mester and M. Felsberg, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 101–110. 11, 12

[21] C. E. García, D. M. Prett, and M. Morari, “Model predictive control: Theory and practice-A survey,” *Automatica*, vol. 25, no. 3, pp. 335–348, may 1989. 37

[22] M. Gombolay, A. Bair, C. Huang, and J. Shah, “Computational design of mixed-initiative human-robot teaming that considers human factors: situational awareness, workload, and workflow preferences,” *The International Journal of Robotics Research*, vol. 36, pp. 5–7, 2017. 12

[23] K. Hornik, “Approximation capabilities of multilayer feedforward networks,” *Neural Networks*, vol. 4, no. 2, pp. 251 – 257, 1991. 74

[24] R. Ikeura, H. Monden, and H. Inooka, “Cooperative motion control of a robot and a human,” in *Proceedings of 1994 3rd IEEE International Workshop on Robot and Human Communication*, jul 1994, pp. 112–117. 11

[25] R. Ikeura, T. Moriguchi, and K. Mizutani, “Optimal variable impedance control for a robot and its application to lifting an object with a human,” in *Proceedings - IEEE International Workshop on Robot and Human Interactive Communication*, 2002, pp. 500–505.

[26] F. Inoue, T. Murakami, and K. Ohnishi, “A motion control of mobile manipulator with external force,” *IEEE/ASME Transactions on Mechatronics*, vol. 6, no. 2, pp. 137–142, jun 2001. 11

[27] A. Q. Keemink, H. van der Kooij, and A. H. Stienen, “Admittance control for physical human–robot interaction,” *The International Journal of Robotics Research*, vol. 37, no. 11, p. 027836491876895, 2018. 12

[28] M. Kennedy, K. Queen, D. Thakur, K. Daniilidis, and V. Kumar, “Precise Dispensing of Liquids Using Visual Feedback,” *IEEE/RSJ 2017 International Conference on Intelligent Robots and Systems, IROS 2017 - Conference Proceedings*, vol. 2017-Septe, pp. 1260–1266, 2017. 13, 88, 93, 99, 105

[29] M. Kennedy, D. Thakur, M. A. Hsieh, S. Bhattacharya, and V. Kumar, “Optimal Paths for Polygonal Robots in SE(2),” *Journal of Mechanisms and Robotics*, vol. 10, no. 2, pp. 21 005–21 008, jan 2018. 11, 12, 52, 53, 54, 58

[30] M. D. Kennedy, L. Guerrero, and V. Kumar, “Decentralized Algorithm for Force Distribution With Applications to Cooperative Transport,” in *Volume 5C: 39th Mechanisms and Robotics Conference*, vol. 5C-2015, no. 57144. American Society of Mechanical Engineers, 2015, p. V05CT08A013. 11, 62, 65

[31] O. Khatib, K. Yokoi, K. Chang, D. Ruspini, R. Holmberg, and A. Casal, “Vehicle/arm coordination and multiple mobile manipulator decentralized cooperation,” *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems. IROS ’96*, vol. 2, pp. 546–553, nov 1996. 11

[32] D. E. Kirk, *Optimal control theory: an introduction*. Courier Corporation, 2012. 35

[33] H. Kohta and M. Ikeda, “Vibration Suppression Control for Mechanical Transfer Systems by Jerk Reduction,” *International Journal of Control, automation and systems*, vol. 5, no. 6, pp. 614–620, 2007. 13

[34] O. Kroemer, E. Ugur, E. Oztop, and J. Peters, “A kernel-based approach to direct action perception,” in *2012 IEEE International Conference on Robotics and Automation*, may 2012, pp. 2605–2610. 13, 98

[35] V. R. Kumar and K. J. Waldron, “Force Distribution in Closed Kinematic Chains Force Distribution in Closed Kinematic Chains,” *IEEE Journal on Robotics and Automation*, vol. 4, no. 6, pp. 657–664, apr 1988. 11

[36] L. Kunze, A. Haidu, and M. Beetz, “Acquiring task models for imitation learning through games with a purpose,” in *IEEE International Conference on Intelligent Robots and Systems*. IEEE, 2013, pp. 102–107. 10

[37] J. D. Langsfeld, K. N. Kaipa, R. J. Gentili, J. A. Reggia, and S. K. Gupta, "Incorporating Failure-To-Success Transitions in Imitation Learning for a Dynamic Pouring Task," in *IEEE/RSJ International Conference on Intelligent Robots and Systems: Workshop on Compliant Manipulation*, 2014. 10

[38] J. D. Langsfeld, K. N. Kaipa, and S. K. Gupta, "Selection of trajectory parameters for dynamic pouring tasks based on exploitation-driven updates of local metamodels," *Robotica*, vol. 36, no. 1, pp. 141–166, 2018. 13, 99

[39] S. Liu, M. Watterson, S. Tang, and V. Kumar, "High Speed Navigation For Quadrotors With Limited Onboard Sensing," *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1484–1491, 2016. x, 55, 56

[40] T. Lopez-Guevara, N. K. Taylor, M. U. Gutmann, S. Ramamoorthy, K. Subr, T. L. Guevara, N. K. Taylor, M. U. Gutmann, S. Ramamoorthy, and K. Subr, "Adaptable Pouring: Teaching Robots Not to Spill using Fast but Approximate Fluid Simulation," *Proceedings of the 1st Annual Conference on Robot Learning*, vol. 78, no. CoRL, pp. 77–86, 2017. 99

[41] M. Macklin, M. Müller, N. Chentanez, and T.-Y. Kim, "Unified Particle Physics for Real-time Applications," *ACM Trans. Graph.*, vol. 33, no. 4, pp. 153:1–153:12, Jul. 2014. xiii, 109

[42] C. Martens, N. Ruchel, O. Lang, O. Ivlev, and A. Gräser, "A FRIEND for assisting handicapped people," *IEEE Robotics and Automation Magazine*, vol. 8, no. 1, pp. 57–65, mar 2001. 9, 13

[43] D. Mellinger, A. Kushleyev, and V. Kumar, "Mixed-integer quadratic program trajectory generation for heterogeneous quadrotor teams," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 477–483, 2012. 12, 52, 55

[44] M. Morari and J. H. Lee, "Model predictive control: Past, present and future," *Computers and Chemical Engineering*, vol. 23, no. 4-5, pp. 667–682, may 1999. 37, 38, 39

[45] R. Mottaghi, C. Schenck, D. Fox, and A. Farhadi, "See the Glass Half Full: Reasoning about Liquid Containers, Their Volume and Content," *Proceedings of the IEEE International Conference on Computer Vision*, vol. 2017-Octob, pp. 1889–1898, 2017. 13, 98

[46] D. Nguyen-Tuong and J. Peters, "Incremental online sparsification for model learning in real-time robot control," in *Neurocomputing*, vol. 74, no. 11, 2011, pp. 1859–1867. 46

[47] ——, "Model learning for robot control: A survey," *Cognitive Processing*, vol. 12, no. 4, pp. 319–340, nov 2011. x, 43, 44, 46

[48] D. Nguyen-Tuong, M. Seeger, and J. Peters, "Model learning with local Gaussian process regression," *Advanced Robotics*, vol. 23, no. 15, pp. 2015–2034, 2009. 46

[49] Y. Noda and K. Terashima, "Simplified flow rate estimation by decentralization of Kalman filters in automatic pouring robot," in *2012 Proceedings of SICE Annual Conference (SICE)*, aug 2012, pp. 1465–1470. 13, 98

[50] ——, “Falling position control of outflow liquid for automatic pouring system with tilting-type ladle,” *IFAC Proceedings Volumes (IFAC-PapersOnline)*, vol. 12, no. PART 1, pp. 53–58, 2007. 13

[51] K. Okada, M. Kojima, Y. Sagawa, T. Ichino, K. Sato, and M. Inaba, “Vision based behavior verification system of humanoid robot for daily environment tasks,” in *Proceedings of the 2006 6th IEEE-RAS International Conference on Humanoid Robots, HUMANOIDS*, dec 2006, pp. 7–12. 10, 12

[52] S. Parikh, V. Grassi, V. Kumar, and J. Okamoto, “Incorporating user inputs in motion planning for a smart wheelchair,” in *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA ’04. 2004*, vol. 2, apr 2004, pp. 2043–2048 Vol.2. 9, 11, 12

[53] P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal, “Learning and generalization of motor skills by learning from demonstration,” in *2009 IEEE International Conference on Robotics and Automation*, may 2009, pp. 763–768. 9, 10, 12

[54] C. J. Phillips, M. Lecce, C. Davis, and K. Daniilidis, “Grasping surfaces of revolution: Simultaneous pose and shape recovery from two views,” in *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2015-June, no. June, may 2015, pp. 1352–1359. xiv, 115

[55] F. Ramos, M. Gajamohan, N. Huebel, and R. D’Andrea, “Time-Optimal Online Trajectory Generator for Robotic Manipulators,” *Eidgenössische Technische Hochschule Zürich, Institute for Dynamic Systems and Control*, p. 6, 2013. 106

[56] A. Rankin and L. Matthies, “Daytime water detection based on color variation,” in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, oct 2010, pp. 215–221. 13

[57] L. Rozo, D. Bruno, S. Calinon, and D. G. Caldwell, “Learning optimal controllers in human-robot cooperative transportation tasks with position and force constraints,” in *IEEE International Conference on Intelligent Robots and Systems*, vol. 2015-Decem, sep 2015, pp. 1024–1030. 10, 11

[58] L. Rozo, P. Jimenez, and C. Torras, “Force-based robot learning of pouring skills using parametric hidden Markov models,” in *9th International Workshop on Robot Motion and Control*. IEEE, 2013, pp. 227–232. 10, 13, 98

[59] S. Sarkka, “On Unscented Kalman Filtering for State Estimation of Continuous-Time Nonlinear Systems,” *IEEE Transactions on Automatic Control*, vol. 52, no. 9, pp. 1631–1641, sep 2007. 42

[60] S. Sastry, *Nonlinear systems: analysis, stability, and control.* Springer Science & Business Media, 2013, vol. 10. 47

[61] C. Schenck and D. Fox, “Towards Learning to Perceive and Reason About Liquids,” in *2016 International Symposium on Experimental Robotics*, D. Kulić, Y. Nakamura, O. Khatib, and G. Venture, Eds. Cham: Springer International Publishing, 2016, pp. 488–501. 13, 98

[62] ——, “Visual closed-loop control for pouring liquids,” in *Proceedings - IEEE International Conference on Robotics and Automation*, vol. abs/1610.0, no. 5, may 2017, pp. 2629–2636. 13, 99

[63] H. Seraji, “A Unified Approach to Motion Control of Mobile Manipulators,” *The International Journal of Robotics Research*, vol. 17, no. 2, pp. 107–118, 1998. 12, 68

[64] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics*. Springer Science & Business Media, 2009. 23, 28, 47

[65] M. W. Spong, S. Hutchinson, M. Vidyasagar, and Others, *Robot modeling and control*. Wiley New York, 2006, vol. 3. 15, 17, 19, 20, 22, 28

[66] R. F. Stengel, *Optimal control and estimation*. Courier Corporation, 1986. 35, 37, 38

[67] J. Stückler and S. Behnke, “Following human guidance to cooperatively carry a large object,” in *IEEE-RAS International Conference on Humanoid Robots*, oct 2011, pp. 218–223. 11, 12

[68] T. G. Sugar and V. Kumar, “Control of cooperating mobile manipulators,” *IEEE Transactions on Robotics and Automation*, vol. 18, no. 1, pp. 94–103, feb 2002. 11

[69] M. Tamosiunaite, B. Nemec, A. Ude, and F. Wörgötter, “Learning to pour with a robot arm combining goal and shape learning for dynamic movement primitives,” *Robotics and Autonomous Systems*, vol. 59, no. 11, pp. 910–922, 2011. 10, 13, 98

[70] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics*. MIT press, 2005. 41

[71] A. Tsiamis, C. P. Bechlioulis, G. C. Karras, and K. J. Kyriakopoulos, “Decentralized object transportation by two nonholonomic mobile robots exploiting only implicit communication,” in *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2015-June, no. June. IEEE, may 2015, pp. 171–176. 11

[72] T. Tsuji and Y. Noda, “High-precision pouring control using online model parameters identification in automatic pouring robot with cylindrical ladle,” in *Conference Proceedings - IEEE International Conference on Systems, Man and Cybernetics*, vol. 2014-Janua, no. January. IEEE, 2014, pp. 2563–2568. 13, 98, 99

[73] L. Vandenberghe, *Convex Optimization*. Cambridge university press, 2009. 31, 32

[74] C.-C. Wang and V. Kumar, “Velocity control of mobile manipulators,” in *[1993] Proceedings IEEE International Conference on Robotics and Automation*, may 1993, pp. 713–718. 12

[75] Y. Wang and S. Boyd, “Fast model predictive control using online optimization,” *IEEE Transactions on Control Systems Technology*, vol. 18, no. 2, pp. 267–278, mar 2010. 37, 38

[76] Z. Wang and M. Schwager, “Force-Amplifying N-robot Transport System (Force-ANTS) for cooperative planar manipulation without communication,” *International Journal of Robotics Research*, vol. 35, no. 13, pp. 1564–1586, 2016. 11

[77] M. Watterson and V. Kumar, “Safe receding horizon control for aggressive MAV flight with limited range sensing,” in *IEEE International Conference on Intelligent Robots and Systems*, vol. 2015-Decem. IEEE, 2015, pp. 3235–3240. 12

[78] D. Williams and O. Khatib, “The virtual linkage: a model for internal forces in multi-grasp manipulation,” in *[1993] Proceedings IEEE International Conference on Robotics and Automation*. IEEE Comput. Soc. Press, 1993, pp. 1025–1030. 11

[79] S. "Xie and Z. Tu, “Holistically-Nested Edge Detection,” in *Proceedings of IEEE International Conference on Computer Vision*, 2015. 108

[80] A. Yamaguchi and C. G. Atkeson, “Stereo vision of liquid and particle flow for robot pouring,” in *IEEE-RAS International Conference on Humanoid Robots*, no. c. IEEE, 2016, pp. 1173–1180. 13, 98

[81] Y. Yamamoto, “Control and coordination of locomotion and manipulation of a wheeled mobile manipulators,” Ph.D. dissertation, Graduate School of Arts and Sciences, University of Pennsylvania, 1994. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.50.2221{&}rep=rep1{&}type=pdf 28, 68

[82] Y. Yamamoto and X. Yun, “Coordinating Locomotion and Manipulation of a Mobile Manipulator,” in *IEEE Transactions on Automatic Control*, vol. 39, no. 6. IEEE, 1994, pp. 1326–1332. 12

[83] ——, “Effect of the dynamic interaction on coordinated control of mobile manipulators,” *IEEE Transactions on Robotics and Automation*, vol. 12, no. 5, pp. 816–824, oct 1996. 12

[84] K. Yokoyama, H. Handa, T. Isozumi, Y. Fukase, K. Kaneko, F. Kanehiro, Y. Kawai, F. Tomita, and H. Hirukawa, “Cooperative works by a human and a humanoid robot,” in *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*, vol. 3. IEEE, sep 2003, pp. 2985–2991. 9, 10, 11

[85] Q. Zang and J. Huang, “Dynamics and Control of Three-Dimensional Slosh in a Moving Rectangular Liquid Container Undergoing Planar Excitations,” *IEEE Transactions on Industrial Electronics*, vol. 62, no. 4, pp. 2309–2318, apr 2015. 13

[86] Y. Zhou, J. Xu, Y. Jing, and G. M. Dimirovski, “Unscented Kalman-Bucy filtering for nonlinear continuous-time systems with multiple delayed measurements,” in *Proceedings of the 2010 American Control Conference*, jun 2010, pp. 5302–5307. 42

[87] Zivkovic and Zoran, “Improved Adaptive Gaussian Mixture Model for Background Subtraction,” in *Proceedings of the Pattern Recognition, 17th International Conference on (ICPR’04) Volume 2 - Volume 02*, ser. ICPR ’04. Washington, DC, USA: IEEE Computer Society, 2004, pp. 28–31. 95