Human motion estimation and controller learning

by

Vladimir Joukov

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2021

Examining Committee Membership

The following served on the Examining Committee for this thesis. The decision of the Examining Committee is by majority vote.

External Examiner: Dr. Sylvain Calinon
Senior Researcher, Idiap Research Institute

Supervisors: Dr. Dana Kulić
Professor, Electrical and Computer Systems Engineering
Monash University

Dr. William Melek
Professor, Mechanical and Mechatronics Engineering
University of Waterloo

Internal Members: Dr. Christopher Nielsen
Associate Professor, Electrical and Computer Engineering
University of Waterloo

Dr. Stephen L. Smith
Associate Professor, Electrical and Computer Engineering
University of Waterloo

Internal-External Member: Dr. James Tung
Associate Professor, Mechanical and Mechatronics Engineering
University of Waterloo

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Humans are capable of complex manipulation and locomotion tasks. They are able to achieve energy-efficient gait, reject disturbances, handle changing loads, and adapt to environmental constraints. Using inspiration from the human body, robotics researchers aim to develop systems with similar capabilities. Research suggests that humans minimize a task specific cost function when performing movements. In order to learn this cost function from demonstrations and incorporate it into a controller, it is first imperative to accurately estimate the expert motion. The captured motions can then be analyzed to extract the objective function the expert was minimizing.

We propose a framework for human motion estimation from wearable sensors. Human body joints are modeled by matrix Lie groups, using special orthogonal groups SO(2) and SO(3) for joint pose and special Euclidean group SE(3) for base link pose representation. To estimate the human joint pose, velocity and acceleration, we provide the equations for employing the extended Kalman Filter on Lie Groups, thus explicitly accounting for the non-Euclidean geometry of the state space. Incorporating interaction constraints with respect to the environment or within the participant allows us to track global body position without an absolute reference and ensure viable pose estimate. The algorithms are extensively validated in both simulation and real-world experiments.

Next, to learn underlying expert control strategies from the expert demonstrations we present a novel fast approximate multi-variate Gaussian Process regression. The method estimates the underlying cost function, without making assumptions on its structure. The computational efficiency of the approach allows for real time forward horizon prediction. Using a linear model predictive control framework we then reproduce the demonstrated movements on a robot. The learned cost function captures the variability in expert motion as well as the correlations between states, leading to a controller that both produces motions and reacts to disturbances in a human-like manner. The model predictive control formulation allows the controller to satisfy task and joint space constraints avoiding obstacles and self collisions, as well as torque constraints, ensuring operational feasibility. The approach is validated on the Franka Emika robot using real human motion exemplars.

Acknowledgements

First, I would like to thank my supervisor Dana Kulić for her support and guidance throughout my graduate studies journey. Your passion for research and dedication to your students is truly inspiring. Thank you for always being available, for organizing so many amazing collaboration opportunities over the years, for the last-minute publication edits, and for your never-ending excitement for novel research directions. Finally, thank you for reviewing my Thesis, editing my endless grammatical mistakes, and committing to a midnight defense.

I also want to thank my doctorate examining committee, Dr. Christopher Nielsen, Dr. Stephen Smith, and Dr. James Tung, for providing insightful feedback and possible future directions, as well as their time to review this Thesis.

This Thesis would not be possible without my research collaborators. I want to thank Dr. Gentiane Venture, Dr. Vincent Bonnet, and all the people in the GVLab for a wonderful exchange experience at the Tokyo University of Agriculture and Technology. Furthermore, my project would not be possible without Dr. Ivan Petrović, Dr. Ivan Marković, and Dr. Josip Cesić, from the University of Zagreb. Thank you for the all the wonderful ideas and an unforgettable visit to Croatia.

This work was only possible due to the constant exchange of thoughts and ideas with all the amazing people in ECE. Thank you, Jonathan Lin, for guiding me at the beginning of my graduate school journey and for the countless hours we have worked together since. Thank you, Kevin Westermann, Pamela Carreno-Medrano, and Brandon DeHart, for always being there to bounce ideas off. Additionally, I am very grateful to have worked with all of the astonishingly bright graduate and post-doctorate students at the Adaptive Systems Lab.

Finally, I want to thank my family and friends, for your unwavering support and understanding throughout this long journey.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Human bodies have evolved to perform complex manipulation and locomotion tasks, capable of efficiently operating under various conditions, which includes carrying light and heavy loads, achieving energy efficient locomotion at various speeds, rejecting and adaptively responding to unknown disturbances, as well as generally adapting to environment constraints. Different aspects of human body capabilities have been a focus of researchers in physiology, anatomy, biomechanics, neuroscience, while also serving as inspiration for humanoid robot design, where robotics researchers aim to develop systems with similar capabilities [2]. The work presented in this thesis aims to bring the capabilities of robots a step closer to those of humans. We take a two fold approach to achieve our goal. First, algorithms are developed to capture human motion using a variety of sensors. Second, we use the estimated movement exemplars to learn the objective functions people utilize when completing a task.

Pose Estimation

To study the human body and its capabilities researchers must first be able to capture and analyze human motion. This is a key enabling technology in many applications, including rehabilitation, athlete performance monitoring, imitation learning and human-robot interaction. When considering applications involving humanoid robots, accurate pose estimation allows the design of controllers to simulate human-like movements through motion re-targeting and imitation learning. In human-robot interaction the participant's pose must be known to guarantee safety and to allow collaborative tasks. To improve the performance of assistive devices in rehabilitation or to enhance user's capabilities with an exoskeleton, the system must be able to first track the motion of the participant [2].

Marker-based motion capture is the gold standard of capturing and analyzing human motion. A set of markers attached to the participant is tracked by multiple cameras to produce the location data of each marker. Assuming the marker placement on the participant is known, the participant's pose is estimated to match the marker positions [3]. These systems allow very precise measurements but require line of sight between the cameras and the markers, can only be used in smaller enclosed spaces, and are costly. Wearable inertial measurement units (IMU) provide a low cost alternative for motion capture. Typically, a sensor consisting of a gyroscope and an accelerometer is placed on each link allowing to track its rotational velocity and acceleration. The measurements are then utilized in accordance with kinematic constraints to estimate human motion. IMU based systems can be used in any environment, do not require external equipment, and are inexpensive.

A major issue in pose estimation from IMUs is gyroscopic drift, which occurs when imperfectly calibrated gyroscope measurements are biased and report non-zero angular velocity while the sensor is stationary. This bias is then numerically integrated into the position data, and is most significant in joint axes that are parallel to the gravity vector, where the measurement of acceleration due to gravity cannot be used to compensate. Another issue is the lack of global position reference, meaning that position data must be obtained by double integrating accelerometer data, which leads to large drift errors as well.

The majority of the current approaches to pose estimation utilizing either motion capture markers or IMUs rely on modeling the human body as rigid segments connected through prismatic or revolute joints [4–7]. For example the ball and socket type joints such as the hips or shoulders are typically modeled with three Euclidean coordinate Euler angles (three perpendicular revolute joints). This modeling approach suffers from gimbal lock, a representational singularity whereby actuating the second joint to 90 degrees brings the first and third joints into alignment and a degree of freedom is lost. Furthermore, it cannot accurately model variance of movement, the same level of noise would cause a very different resulting motion based on how close the configuration is to the representational singularity. To enable more accurate measurement of human pose and its variability, representations that do not introduce a representational singularity are preferred [8].

Pose estimation accuracy can be improved by incorporating state constraints [1]. Consider a collaborative box carrying task between a robot and a human. Since both are holding the box, the possible poses they could be in are restricted. Incorporating knowledge of the task constraints allows for higher accuracy pose estimation. Constraints are also useful if the pose estimation is used to provide visual feedback; when capturing human motion for animation or interacting with a virtual environment it is desirable to maintain accurate visualization of interaction with virtual objects. Human motion is constantly constrained through multiple interactions with the environment. In common activities of daily living, people are constantly in contact with the ground when standing and walking, with the environment when touching light switches, tools and

appliances, and with other people in handshaking or hand-off tasks. Incorporating knowledge of these constraints into the estimation can improve the accuracy by preventing drift artifacts from pushing the pose estimation though contact points. Current constrained pose estimation approaches assume that the active constraints are known *a-priori* [9]. However, as the person interacts with their environment, the constraints are continuously changing. Thus for accurate pose estimation it is important to only enforce the currently active constraints.

Objective Learning

After accurately tracking human motion we would like to transfer it to robotic devices so they can also perform natural human like motion. However, it is not enough to simply follow the captured human exemplar trajectories. Simply replaying human walking trajectories on a full humanoid robot will not allow it to walk, it must also account for its own kinematic and dynamic constraints, reject disturbances, and handle collisions.

Thus, in order to transfer human like movement characteristics to robots or active assistive devices it is important to extract and understand the underlying objective functions people utilize and re-target these to the machine. Such an approach can then not only follow a predefined trajectory but also react to disturbances in a human like fashion while satisfying its own dynamical constraints. Furthermore, for the controller to be utilized in assistive devices or human robot interaction applications it must handle constraints to ensure safety of the human participants.

Historically, robot behaviours are manually designed and tuned for each robotics task. Instead, teaching robots by demonstration offers a promising way to eliminate the time intensive, tedious, manual programming required for even simple movement tasks and would allow robots to learn motions from humans who are expert at a specific task without requiring them to have robotics knowledge, which could lead to better, human like motion [10]. Furthermore, learning from demonstration may capture elements that are inherently difficult to manually program such as intent of the motion or emotion associated with it [11].

Current approaches to learning objective functions from demonstrations typically assume a quadratic cost function as a linear combination of predefined basis functions such as joint velocities, accelerations, jerk, and torques [12, 13]. Coefficients of the basis functions are then optimized such that the motion reproduced by the controller minimizing the objective function matches the exemplars. This results in an objective function with predefined basis, constant weights, and a large number of parameters that must be optimized. Furthermore, it does not capture the variance in human motion.

1.1 Contributions

The research presented in this thesis develops techniques for accurate on-line motion estimation using various modalities, and learning methods to enable the transfer of human motion strategies to robots. In order to overcome limitations of current pose estimation approaches, we propose a novel method that can be used with a variety of sensing sources, including motion capture markers and wearable inertial measurement units. The approach performs stochastic inference of human motion by defining the state space to reside on a Lie group, with each state element corresponding to the kinematic model of the analyzed human body part. The method is shown to have a better human motion variance representation in spherical joints and is not effected by gimbal lock.

An additional approach for pose estimation is developed that is capable of handling constraints. The constraints greatly improve the estimation results in cases such as object handling and contact with the environment. By automatically identifying the most likely active constraint the approach can accurately estimate human gait including the global body position utilizing only wearable IMUs.

Next, re-targeting human motion onto robotic manipulators is tackled. A new multi-output approximation of Gaussian processes is derived that leads to orders of magnitude computational improvement for both training and regression and allows Gaussian processes to be utilized in real time control. Movements are modeled using this new method capturing variability and correlations in human motion exemplars. The model is then utilized as a novel non-parametric method to learn the control objective from demonstration. The learned controller incorporates variance of the expert exemplars, does not assume any specific structure, and can be applied to any manipulator in task or joint space.

The thesis makes the four following contributions to the state of the art in human pose estimation and manipulator control:

Human motion estimation on Lie groups:
The first contribution is a framework for human pose estimation from wearable sensors that relies on Lie group representation to model the geometry of human movement. Human body joints are modeled by matrix Lie groups, which are a type of manifold often used in physical sciences and engineering. The joints orientations are described using special orthogonal groups SO(2) and SO(3) for joint pose, depending on the number of degrees of freedom of an associated joint, and special Euclidean group SE(3) for base link pose representation. The human body system is assumed to propagate following a constant acceleration motion model. To estimate the human joint pose, velocity and acceleration, the equations for employing the Extended Kalman Filter

on Lie Groups (LG-EKF) are developed, to explicitly account for the non-Euclidean geometry of the state space. The LG-EKF observation model for articulated motion estimation based on marker position, gyroscopic and accelerometer measurements is derived. The observability of an arbitrarily long kinematic chain of SO(3) elements is characterized, based on a differential geometric approach. The proposed algorithm is compared to two competing approaches, the EKF and unscented KF (UKF) based on Euler angle parametrization, in both simulations and extensive real-world experiments. The results show that the proposed approach achieves significant improvements over the Euler angle based filters. It provides more accurate pose estimates, is not sensitive to gimbal lock, and more consistently estimates covariances.

Constrained human motion estimation:
Wearable inertial measurement unit sensors are often used for real time pose estimation due to their low cost and suitability for both indoor and outdoor environments. However, most approaches assume that the body movement is unconstrained and suffer from position drift. Incorporating constraints such as contact with the environment can greatly improve estimation accuracy and allow for global position tracking. An estimation framework that takes into account interaction constraints is proposed. The approach can handle constraints with respect to the global (environment) or local (self) frames and automatically identifies the currently active constraints. The method is extensively validated in simulation and with human motion data and is shown to accurately track both the joint angles and the global body position during gait, resulting in less than 5% Cartesian error relative to distance travelled.

Fast approximate multi-output Gaussian processes (FAMGP):
Gaussian processes regression models are an appealing machine learning method as they learn expressive non-linear models from exemplar data with minimal parameter tuning and estimate both the mean and covariance of unseen points. However, exponential computational complexity growth with the number of training samples has been a long standing challenge. During training, one has to compute and invert an $N \times N$ kernel matrix at every iteration. Regression requires computation of an $m \times N$ kernel where N and m are the number of training and test points respectively. This thesis shows how approximating the covariance kernel using eigenvalues and functions leads to an approximate Gaussian process with significant reduction in training and regression complexity. Training with the proposed approach requires computing only a $N \times n$ eigenfunction matrix and a $n \times n$ inverse where n is a selected number of eigenvalues. Furthermore, regression now only requires an $m \times n$ matrix. Finally, in a special case the hyperparameter optimization is completely independent from the number of training samples. The proposed method can regress over multiple outputs, estimate the derivative of the regressor of any order, and learn the correlations between them. The computational complexity reduction, regression capabilities, and multi-output correlation learning are demonstrated in simulation examples.

FAMGP based human objective function learning and control:

Learning from human demonstrations can enable robots to quickly acquire new skills, generate human like motion, and allow novice users to teach robots intricate tasks. Research suggests that humans minimize a task specific cost function when performing movements. This thesis presents a novel method to learn a time varying cost function from demonstrations using fast approximate Gaussian processes. The learned cost function is then optimised in a model predictive control framework to reproduce the task on a robot while satisfying constraints and minimizing joint torques. The proposed approach accurately encodes variability of the expert motion as well as correlations in task or joint space. This allows the robot to understand what parts of the task to focus on and react in a human like way to disturbances. Re-parameterizing the model in terms of a phase variable allows exemplar alignment for training and a controller that can handle encountering obstacles. The proposed approach is compared in simulation to two other popular trajectory distribution modeling methods and is extensively tested on the Franka Emika robot showing its ability to handle disturbances, constraints, obstacles, and reproduce human demonstrated motions.

1.2 Outline

This thesis is structured as follows: In the second Chapter the relevant literature is reviewed. Chapter 3 presents the necessary mathematical background for the formal development of our proposed motion estimation and learning approaches. Chapter 4 develops the Lie group based pose estimation approach using motion capture markers or wearable inertial measurement units. Chapter 5 improves wearable IMU based pose tracking by incorporating constraints into the estimation process. In chapter 6 we shift towards learning the underlying control strategies from estimated motion and develop a novel approximation for Gaussian processes improving training and regression computational requirements by orders of magnitude. Chapter 7 describes how this approximate Gaussian process can be used to learn a quadratic time varying cost function from demonstrations which is utilized in a model predictive control framework to reproduce human like motions on a manipulator. We conclude in Chapter 8 summarizing the presented work and discussing future research directions.

Chapter 2

Related Work

This chapter reviews the current state of the art approaches to human pose estimation and imitation learning. First, we present the benefits and drawbacks of optical marker and inertial sensor based motion capture, focusing on Inertial Measurement Unit gait estimation since it has a very wide range of practical applications. The limitations of the commonly used Euclidean representation human kinematic model is discussed. Next, imitation learning is differentiated into two overarching approaches, trajectory and controller based learning. We discuss the differences between the two approaches and present the latest advancements and their limitations.

2.1 Motion Estimation

A number of different sensing modalities have been proposed for human motion measurement. The proposed methodologies can be split into approaches relying on wearable sensors, such as accelerometers, inertial measurement units (IMUs), or contact switches and ones with sensors placed in the environment, using cameras, LIDAR, and beacons. While the former provide a cheaper and more portable solution, they are usually significantly less accurate. Optical motion capture is a method to record the movements from body worn markers observed by multiple cameras. The 3D positions of the markers are extracted from the images using the relative positions of the cameras to each other and are analyzed to compute the pose [14]. It is considered the gold standard for human motion measurement and is widely used by biomechanics, kinesiology, and robotics researchers. However, when line of sight between the sensor and the human cannot be ensured, when motion is to be captured in large or outdoor spaces, or when cost is an issue, wearable sensing using IMUs is preferred [15].

2.1.1 Marker based pose estimation

When relying on marker position measurements, a kinematic model of the participant is typically defined based on anthropomorphic tables or by measurements of markers' positions, which are assumed to be rigidly attached to the skeleton links. Unfortunately, for a full body skeletal model, there is no closed form solution for the inverse kinematics (IK), hence differentiation can be used to iteratively solve for joint angles using the pseudoinverse of the Jacobian. In singular configurations the Jacobian is not invertible and then it is possible to include a non-zero damping constant in the least squares minimization to maintain full rank [16]. The Jacobian inverse based methods do not account for stochastic error in marker position measurements, are greatly affected by outlier measurements, and are not capable of predicting future poses. By treating the skeleton pose as a state and 3D marker positions as measurements, recursive stochastic estimators can be used to help reduce the effect of stochastic marker position errors. Including the joint positions, velocities, and accelerations in the motion model of the filter helps to maintain correct pose estimate during short term occlusions. Various stochastic filters have been proposed for IK, such as the Smart Sampling Kalman Filter [4] and the Unscented Kalman Filter [5]. Beyond pose estimation, filtering approaches can also be used for jointly estimating pose and dynamic parameters of a body [6].

In recent years, as more computational power has become available, multiple optimization based approaches have been proposed. Meyer *et al*. used an expectation minimization algorithm to jointly estimate marker labels and the pose on-line [17]. For each new frame of marker positions, they optimize the assignment of marker labels and the skeletal configuration, using an expectation maximization algorithm and relying on numerical differentiation. An additional joint limit cost function attempts to avoid exceeding joint limits such as the knee bending backwards. While the algorithm is able to run online for a single participant, the computational complexity of numerical differentiation and optimizing for each frame will quickly become intractable as the number of models and markers increases in the scene. Furthermore, since each joint is modeled as a quaternion and can achieve arbitrary orientation, there is no guarantee that the estimated pose will meet human joint constraints. When online estimation is not necessary, optimization approaches can utilize the entire dataset and include not only pose but skeletal model parameters as well. Ayusawa *et al*. used nonlinear programming to simultaneously perform inverse kinematics and estimate the geometric parameters of the model [18]. By including force plates as additional measurements to the marker positions it is possible to optimize over joint positions, velocities, and accelerations, as well as the geometric and inertial body segment parameters [7]. The approach models the ankle, knee, hip, and shoulder joints as hinges and is only capable of estimating motion in the sagittal plane. Optimization methods aim to minimize the distance between modeled and actual marker positions. However, they do not account for the presence of noise in the measured

marker positions. This can lead to unrealistic jittery estimated motion.

2.1.2 Inertial Measurement Unit based pose estimation

Inertial measurement units (IMUs), small body worn sensors that measure accelerometer and gyroscope signals, are a popular method for human motion tracking. IMU-based pose estimation does not suffer from line of sight or cost constraints of vision-based or motion capture-based techniques, and can be used in both indoor and outdoor environments. Due to these advantages, they have been deployed in numerous human motion tracking settings in healthcare, biomechanics, fitness, activity recognition, imitation learning, and human-robot interaction [19, 20].

IMU based pose estimation methods typically falls into one of several categories: (1) utilizing the accelerometer as an inclinometer [21, 22] to estimate motion with respect to the gravity vector, (2) combine accelerometer data, which is accurate in slow or stationary movements, with gyroscope data, which is accurate in fast movements but is prone to drift error [23, 24], or (3) use the IMU data in a sensor fusion framework, such as the extended Kalman filter (EKF). EKF utilizes a kinematic model to estimate the expected sensor values. The joint angles in the kinematic model are then updated to minimize the estimated sensor error between the estimated and the measured values [1, 25, 26]. However, gyroscope angular drift and accelerometer positional drift are both significant problems that EKF reduces but does not eliminate.

Positional constraints are a potential solution to this problem, by including position information into the pose estimation. Existing methods to introduce constraints into IMU-based joint angle pose estimation algorithms tend to fall into two categories: (1) adding a second sensor modality, or (2) making a simplifying assumption about the nature of the movement.

Sensors such as cameras [27–29], contact and pressure sensors [30–32], flex sensors [33], or magnetometers [34] have previously been incorporated into pose estimation to improve the estimation accuracy. Obtaining gait events such as toe-off or heel-strike by pressure sensors is also a common approach [35–37]. However, these additional modalities impose additional environmental requirements, such as requiring line of sight or a lack of magnetic distortion, and increase the technology complexity, which may limit the system's applicability.

Instead of extra sensors, constraints can be introduced in the form of external information formed by *a priori* knowledge about the nature of the motion. These can include knowledge about the human range of motion [1], or about the cyclic nature of the task [38]. In a previous EKF based estimation approach [39], we applied thresholds on the ankle accelerometer signal to detect contact changes and manually switched the base of a kinematic chain from left to right foot to estimate straight line gait.

2.1.2.1 IMU-based Gait Pose Estimation

IMUs have often been used for gait tracking, with two classes of approaches: (1) methods that estimate stride length by estimating the occurrence of gait toe-off and heel-strike events, as well as (2) methods that estimate the leg joint angles.

A simple thresholding approach on IMU signals can be used to detect constraint events such as toe-off and heel-strike, and estimate the distance travelled [40]. Contact can be detected using thresholds and zero crossings on the IMU signals and time elapsed [22, 26, 39, 41–43]. Distance traversed or ankle joint angle can be estimated by assuming a simplified gait such as the pendulum gait [44, 45] or integrating along the axes of movement [46–48]. Trojaniello *et al*. [49] demonstrated that different IMU threshold-based techniques generally report similar stride time. However, these methods only provide the distance travelled, often only in a single direction, and do not provide joint angle estimates.

A few approaches estimate the joint angles of the hip, knee, and ankle using IMUs. These methods tend to treat each limb independently and calculate the orientation of the sensor, and thus the limb it is attached to, individually, using the complementary filter [22, 50–52], Kalman filter [34, 53], machine learning [54], or employ a kinematic chain with the hip or ankle fixed in space [39, 55]. These methods however do not model free-body prismatic movement of the body and assume transitional acceleration to be noise.

To provide both Cartesian position and joint angle estimates during gait, Sy *et al*. [9] used IMU EKF to estimate the motion of each link separately and applied kinematic, foot position, and pelvis constraints in post processing. Pelvis position is determined by double integration of the accelerometer data, and corrected by a virtual pelvis position sensor to keep the pelvis close to average position of the two ankles, and at the height of the unbent leg. However, this method assumes steps only occur on flat surfaces, consistent torso height and zero ankle velocity of the stance foot. It also requires additional tuning to limit the position error covariance from growing without bounds. The approach was validated against motion capture using gait motion and obtained a floating base position error of 0.05 m, orientation error of 16°, and average sagittal joint error of 10°. In a similar fashion, Miezal *et al*. [56] estimate the motion of each segment separately and include a zero contact point velocity and zero contact point height pseudo measurements. The method utilizes the height of each possible contact point to compute a probability that it is in contact with the ground and applies the pseudo measurement if contact is likely. The approach was validated on a single participant and achieves a global pelvis translation error of 0.15m on average.

Thus, there are multiple IMU-based methods that aim to extract specific metrics such as gait event timing but do not provide joint angle estimates and therefore cannot capture the intricacies

of human motion. When estimating joint angles a popular approach is ignoring the global body position and assuming fixed base kinematics. These methods cannot estimate the path of the walker or the total distance walked which are commonly used for gait assessment. Prior works that propose incorporating constraints to allow position tracking typically require an additional sensor, separate step detection algorithm, or make restricting assumptions about the motion being tracked such as consistent pelvic height, which may limit the generalizability of the method.

IMUs provide an excellent alternative to camera based motion capture however so far, due to the lack of a direct position measurement, the accuracy has not been comparable. The addition of constraints can greatly improve the pose estimate but the current methods rely on knowing the constraints a priori, assuming the constraint is always active, utilizing additional sensors, or additional task specific constraint detection algorithms. An approach that can handle both within model as well as environment interaction constraints and detect when a constraint becomes active can further improve IMU based pose estimation leading to a low cost system applicable in any environment.

2.1.3 Pose representation

In the aforementioned methods the kinematic models are rigid links connected with joints that may be revolute, translational, or spherical. Often the state of each joint is represented by the position, velocity, and acceleration. To capture human motion we need to estimate this state based on available measurements. Prismatic joints in one direction can easily be described by a single variable. However, there are multiple possible representations for revolute and spherical joints as well as for arbitrary three dimensional transformations. All of the aforementioned works utilize representations of transformations in the Euclidean space. While [57] uses a quaternion joint representation, the approach cannot represent different constraints for human joints with different degrees of freedom (dof): 3 at the hip and shoulder, 2 in the elbow and wrist, and a single dof at the knee. However, human motion and many other types of motion of interest in robotics do not occur in Euclidean space, but rather arise on curved geometries often called manifolds. By using the manifold representations, the overall performance of wide variety of applications can be significantly improved [58, 59]. Among manifolds, the representation of pose and attitude arise most commonly in robotics applications. In particular, the attitude of an object can be modelled as a special orthogonal group $\mathrm{SO}(n)$, $n = 2, 3$, while the pose can be modelled as a special euclidean group $\mathrm{SE}(n)$, $n = 2, 3$ [58]. Notably, both $\mathrm{SO}(n)$ and $\mathrm{SE}(n)$ are examples of matrix Lie groups. A number of studies have investigated the uncertainty representation and association to Lie groups, such as $\mathrm{SO}(3)$ [60], $\mathrm{SE}(2)$ [61], $\mathrm{SE}(3)$ [58]. Recently, several theoretically rigorous approaches for filtering on Lie groups have been proposed. In [62] the authors proposed an EKF able to perform estimation respecting the geometry of matrix Lie groups, the unscented

transform-based filtering approach was proposed in [63] and the particle-based approach was presented in [64]. It is also possible to represent SO(3) and SE(3) elements using regular and dual quaternions respectively. Sabatini proposed an EKF formulation to estimate rotation, represented as a quaternion, and sensor bias for a wearable IMU [65]. Rangaprasad *et al*. [66] used dual quaternions to represent elements of SE(3) and derived a linear measurement model for pose and position measurements.

The benefit of manifolds for human action recognition has already been explored in the literature. Lui has proposed an algorithm for action recognition by relying on a description on Grassmann manifolds [67]. In [68] the authors exploited the manifold structure by relying on the particle filter for learning purposes, while in [69] the authors use different manifolds as priors for manifold learning. Devanne *et al*. have used a spatio-temporal modeling of trajectories in a Riemannian manifold for action recognition purposes [70]. Recently, Brossette *et al*. have proposed the posture generation problem that encompasses non-Euclidean manifolds as well [71]. However, few works have modeled human joints as Lie group elements and estimated full body pose while respecting the geometry. Applying the aforementioned filtering methods to capture human motion on the group can lead to better motion uncertainty representation and improved tracking accuracy.

2.2 Imitation Learning

Over two thousand years ago Aristotle stated "Imitation is natural to man from childhood, one of his advantages over the lower animals being this, that he is the most imitative creature in the world, and learns at first by imitation". Current research supports this claim [72], showing that infants imitate facial and manual gestures [73] and athletes rely on observing demonstrations to learn new skills and strategies [74, 75]. Research in neuroscience also identifies brain regions dedicated to imitation learning, showing a connection between perception and motor systems and finding that the same "mirror neurons" activate both when observing a movement and performing it [76]. Athletic coaches consider demonstration to be a key factor in teaching motor skills [77] and interviews of educators and students suggest that both view it as a valuable teaching tool [78], suggesting that demonstrating is the preferred method of motor skill transfer. Apart from learning from demonstration, it have been shown that children prefer to teach a motor tasks by demonstration rather than verbal description and athletic coaches consider demonstrating movement a valuable and effective teaching tool [77].

The evidence that imitation learning is innate to humans led researchers to consider its possible application in robotics. Most robots today are programmed by operators who use their understanding of a specific task to create the desired behavior. This limits the robot to situations

the human operator has considered and does not permit easy adoption of new tasks [79]. Teaching robots by demonstration offers a promising way to eliminate the time intensive, tedious, manual programming required for even simple movement tasks and would allow robots to learn motions from humans who are expert at a specific task without requiring them to have robotics knowledge, which could lead to better, human like motion [10]. Furthermore, learning from demonstration may capture elements that are inherently difficult to manually program such as intent of the motion or emotion associated with it [11]. An ideal imitation learning algorithm would learn from few repetitions, capture the expert's focus throughout the task, and generalize between robots.

Approaches to imitation learning can be separated into trajectory and controller based categories. The former focuses on learning a model of the trajectory in either joint or task space. Once a model is learned, human-like trajectories can be generated and tracked by robots utilizing classical control methods. The latter learns the controller used by the demonstrator. A manipulator employing the learned controller should then automatically produce human like motion [80].

2.2.1 Movement primitives

A popular trajectory based method is to encode the expert demonstrations using dynamic movement primitives (DMPs) [81]. DMPs can be considered as a stable point attractor dynamical system with an additional learned forcing term, modeled as weighted basis functions, that decays to zero as the goal point is reached. Regular DMPs have been used to teach robots to play table tennis [82], locomotion [83], and grasping [84]. While DMPs are very good at encoding trajectories with few parameters and allow for easy adaptations such as scaling or changing the goal points, they cannot capture the variability in the expert exemplars or the correlation between state elements. Consider a simple task of reaching for a cup. During most of the reaching motion, there may be high variability in the hand position because it is not particularly important. However, as the hand gets closer to the cup, it must grasp the cup by the handle in a particular orientation and the variability becomes very low. Thus capturing the expert variability can lead to learning what is important in a task. The human musculoskeletal model as well as neural control creates correlations that can be seen in joint or task space [85]. Since the robot dynamics do not match the musculoskeletal model, the motion and effect of disturbances would appear very different. Thus, learning the correlation between state elements from the human exemplars can lead to much better human like motion reproduction. To address this issue, DMPs have been extended to probabilistic movement primitives (ProMPs) [86] that can model trajectory distributions. ProMPs learn the basis function weights for each trajectory exemplar and then place a Gaussian prior on the weights by computing their mean and covariance. ProMPs have been applied to teach robots many independent and collaborative tasks. Gomez *et al*. [87] adapted learned primitives in real

time to ensure the incoming ball is struck with the paddle at a specific position, orientation, and velocity. Collaboration using ProMPs is possible by learning the correlation between multiple agents (human and robot) [88]. While one can deduce the correlation between state variables modeled by ProMPs it is not captured directly, only through the distribution of the basis function weights.

Another popular approach that can capture the variability and correlation in the data are Gaussian Mixture Models (GMMs). GMMs represent the trajectory using a set of Gaussians including the correlation between state and input variables. Typically joint or task space coordinates and time can used respectively. Provided multiple trajectory exemplars the means and covariances of the Gaussians can be found offline using an optimization algorithm [89]. After the model is trained, Gaussian mixture regression (GMR) can be used to efficiently predict the state vector given an input. Since Gaussians retain their properties under linear transformations it is possible to consider the same exemplar data from different frames of reference. This allows the approach to generalize to novel situations *i.e.* by looking from the end effector and target placement frames, object placement can be performed regardless of the world frame object positions and orientation present in the training data set [90]. In a similar way to GMM, Hidden Markov models (HMM) can be used to provide a stochastic model of the trajectory. HMM encodes the state distribution via Gaussians at specific key points in the trajectory and the probabilities of transitioning between them. Then GMR can be used to predict the trajectory while taking into account the probability of the state sequence [91]. While these approaches can capture the variability and correlation, they require learning means and covariances of multiple multivariate Gaussians to represent the data. As the complexity of the motion grows so does the necessary number of Gaussians and thus the computational complexity of training and inference.

2.2.2 Distribution based control

Trajectory based methods may model the distribution of human movements, but how this information is used is left up to the controller. Optimization methods typically use a fixed cost function for the entire duration of the trajectory and do not consider the variance in human motion. To incorporate probabilistic modeling into the controller, Englert *et al*. used Gaussian Processes (GPs) to model manipulator dynamics and expert exemplars [92]. The control policy is then learned as a radial basis function network such that the Kullback-Leibler divergence is minimized between the learned GPs. This approach trains a robot specific controller and does not incorporate task or joint space constraints. Calinon used Gaussian Mixture Models (GMM) to build task-parameterized models and showed how they can be utilized in a model predictive control framework to generate desired accelerations [90]. Building the GMM from multiple frames of reference leads to good generalization of the task, however the double integrator

controller formulation does not explicitly take into account the manipulator's torque limits and dynamics.

2.2.3 Inverse optimal control

Despite the many degrees of freedom in the human body, for a specific task humans typically use a very small subset of the possible motions. Many theories of motor function are based on optimal performance, hypothesizing that, for a given task, the person is minimizing some internal cost function [93]. It has been shown experimentally that optimal feedback control can accurately model human movements in reaching tasks [93], industrial screwing tasks [94], and gait [95]. Since quality of sensory information can be dependent of the motion, such as turning your head to look at an object, humans incorporate feed forward control strategies in order to take sensory costs into account [96]. The learned cost functions humans are minimizing during their movement can then be used on a robotic device with a controller minimizing the objective while handling the dynamic and kinematic constraints of the system.

When learning the human objective function it is often assumed that the expert is minimizing some integral cost function. If the cost is a linear combination of known basis functions it is possible to estimate their coefficients using bi-level optimization which consists of coefficient estimation on the upper level and constrained optimal control problem on the lover level [12]. A key drawback of this approach is the assumption that the basis functions are known and their coefficients remain constant throughout the entire motion. Furthermore, the bi-level approach is computationally demanding, making it difficult to use with large exemplar data-sets. By assuming the exemplar trajectories are optimal and employing the Karush-Kuhn-Tucker (KKT) necessary optimality conditions, it is possible to simplify finding the coefficients to a least squares problem [13]. If the trajectories are not well represented by the assumed cost functions, the KKT Jacobian may be rank deficient and thus not invertible. Furthermore, to keep the inversion computationally tractable, the KKT method does not use the entire trajectory but it's approximation with splines, this may lead to learning a controller which does not capture smaller details in the exemplar trajectories.

2.2.4 Inverse Reinforcement Learning

An alternative approach for learning a controller from demonstration is based on *inverse reinforcement learning* (IRL). In the reinforcement learning framework, an agent occupies a state in the environment and performs an action to move to another state. The environment is typically represented by a Markov decision process (MDP) defined by a set of states, actions, a state

transition model describing how the agent transitions from state to state as a consequence of its actions, and a reward model that describes the desirability of states and actions. Reinforcement learning, through interaction between the agent and its environment, finds the optimal policy which maps states to actions such that the reward is maximized [97]. Inverse reinforcement learning (IRL) is the problem of finding the reward function given observations of an agent following the optimal policy [98]. In large state spaces it becomes intractable to learn a reward for each state action pair. Similarly to inverse optimal control, the problem can be made tractable by imposing a structure on the reward function, for example as a linear combination of state dependent feature vectors [98]. To overcome limitations of linear structures, reward functions modeled by deep neural networks [99] and Gaussian processes have been proposed [100]. One drawback of IRL is reward function multiplicity, there may be multiple reward functions for the same optimal policy [101].

In summary, learning from demonstration is approached in multiple different ways with the end goal of capturing human expert motion and re-targeting it to a robot. Current trajectory based methods capture the movement but may not handle disturbances in a human like manner. Inverse optimal control and reinforcement learning methods assume that the objective or reward functions can be parameterized by a small set of variables and use optimization to learn them. Selecting the relevant parameters remains an open problem, furthermore, there is no guarantee that optimization converges to the global minimum. Other machine learning methods such as deep neural networks may produce a good model but are computationally expensive, requiring extremely large data-sets for training, that are time consuming to collect, and not achieving fast enough prediction for real time control. Distribution based approaches are a promising way to model human motion, capturing both the mean and variance. However, as the complexity of the motion grows so do the number of modeling parameters making the current approaches difficult to utilize on real manipulators. A non parametric approach that can learn the expert objective function from only a few examples and provide fast inference to be utilized in control applications can be of great benefit.

Chapter 3

Mathematical Background

This chapter provides the necessary mathematical background to develop the algorithms in later sections of the Thesis. We begin by introducing Lie groups and algebras in section 3.1 which are used in chapter 4 as a better representation of human joints than typical Euclidean approaches. Next, the Kalman filter and its extensions are discussed in Section 3.2, it is the chosen stochastic state estimator in chapters 4 and 5. Section 3.3 introduces Gaussian processes and its benefits and limitations as a machine learning tool. In chapter 6 we derive an approximate Gaussian process which significantly improves computational efficiency over the standard approach and chapter 7 relies on this approximation to learn objective functions from human demonstrations. Finally, Section 3.4 reviews linear model predictive control which we utilize in the manipulator control framework presented in chapter 7 to reproduce human like motions.

3.1 Lie groups and Lie algebra

We now introduce the concept of Lie groups and Lie algebra as prerequisites for estimation on Lie groups. A Lie group G is a group which also has the structure of a smooth manifold. The group operators, composition and inversion, are smooth operations. Each point $X \in \mathrm{G}$ has an associated tangent space $T_X(\mathrm{G})$ [102]. This linear tangent space is usually placed at the group identity, and is called the Lie algebra of G, which we denote by $\mathfrak{g}$ [103]. The Lie algebra $\mathfrak{g}$, which is of the same dimension as G, admits a binary operation $[\cdot, \cdot]$ called the Lie bracket, which reflects the non-commutative content of the group operation. Furthermore, if the group G is a matrix Lie group, then $\mathrm{G} \in \mathbb{R}^{n \times n}$ and group operations are simply matrix multiplication and inversion.

The Lie algebra $\mathfrak{g} \in \mathbb{R}^{n \times n}$ associated to a p-dimensional matrix Lie group $\mathrm{G} \in \mathbb{R}^{n \times n}$ is a p-dimensional vector space defined by a basis consisting of p real matrices E_r, $r = 1, \ldots, p$,

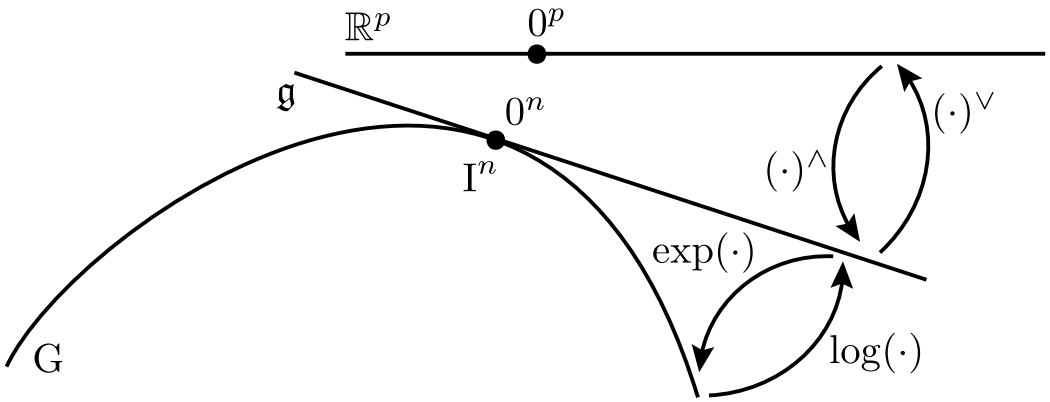


Figure 3.1 An illustration of mappings within the triplet of Lie group G, Lie algebra G and the Euclidean space $\mathbb{R}^p$.

often referred to as generators [104]. In particular, a Lie algebra is an open neighbourhood around $\mathbf{0}^p$ in the tangent space of G at the identity I^n. The matrix exponential $\exp_{\mathrm{G}}$ and logarithm $\log_{\mathrm{G}}$ establish a local diffeomorphism as

$$\exp_{\mathrm{G}} : \mathfrak{g} \to \mathrm{G} \quad \text{and} \quad \log_{\mathrm{G}} : \mathrm{G} \to \mathfrak{g}. \tag{3.1}$$

A natural relation between the p-dimensional Lie algebra G and the Euclidean space $\mathbb{R}^p$ is given through a linear isomorphism

$$[\cdot]^\vee_{\mathrm{G}} : \mathfrak{g} \to \mathbb{R}^p \quad \text{and} \quad [\cdot]^\wedge_{\mathrm{G}} : \mathbb{R}^p \to \mathfrak{g}. \tag{3.2}$$

An illustration of the above mappings is given in Fig. 3.1.

For brevity, we will use the following notation [105]

$$\exp^\wedge_{\mathrm{G}}(x) = \exp_{\mathrm{G}}([x]^\wedge_{\mathrm{G}}) \quad \text{and} \quad \log^\vee_{\mathrm{G}}(X) = [\log_{\mathrm{G}}(X)]^\vee_{\mathrm{G}}, \tag{3.3}$$

where $x \in \mathbb{R}^p$ and $X \in \mathrm{G}$.

Since Lie groups are generally non-commutative, i.e., $XY \neq YX$, we also need to employ the adjoint representations. The adjoint representation of G on $\mathfrak{g}$, Ad_{G}, can be interpreted as of representing the elements of the group as a linear transformation of the group's algebra, and in general, it measures the failure of $X \in \mathrm{G}$ to commute with elements of G near the identity [106]. The adjoint representation of G, ad_{G}, is the differential of Ad_{G} at the identity and for a commutative group ad_{G} evaluates to zero.

3.1.1 Concentrated Gaussian distribution

To make use of stochastic filtering on Lie groups we need a notion of a Gaussian distribution on Lie groups. A distribution on a Lie group that is tightly focused, meaning that almost all the mass of the distribution is concentrated in a small neighborhood around the mean, can be expressed in the Lie algebra [58, 107], and is called a concentrated Gaussian distribution (CGD). Let $X \in \mathrm{G}$ be a CGD random variable with mean μ and covariance P as

$$X = \mu \exp_{\mathrm{G}}^{\wedge}(\epsilon), \quad X \sim \mathcal{G}(\mu, P), \tag{3.4}$$

where $\epsilon \sim \mathcal{N}_{\mathbb{R}^p}(\mathbf{0}^p, P)$ is a zero-mean Gaussian distribution with covariance $P \subset \mathbb{R}^{p \times p}$ defined in the Lie algebra, i.e., the Euclidean space $\mathbb{R}^p$. We can see from (3.4) that the mean value μ is defined on G, while the associated uncertainty resides in $\mathbb{R}^p$. This concept allows us to work with the covariance in $\mathbb{R}^p$ and use Euclidean tools, as we would with a 'classical' Gaussian distribution [62]. For a more formal introduction of the concepts presented here, the reader is referred to [108].

3.1.2 Special orthogonal group $\mathbf{SO}(2)$

The $\mathrm{SO}(2)$ group represents a rotation around a single axis:

$$\mathrm{SO}(2) = \left\{ X \in \mathbb{R}^{2\times 2} \mid X^{\mathrm{T}} X = I, \, \det(X) = 1 \right\} . \tag{3.5}$$

For a single degree of freedom rotation representation consisting of an angle $x = \phi$, the Lie algebra $\mathfrak{so}(2)$, is given as

$$x^{\wedge}_{\mathrm{SO}(2)} = \begin{bmatrix} 0 & -\phi \\ \phi & 0 \end{bmatrix} \in \mathfrak{so}(2) . \tag{3.6}$$

where $(\cdot)^{\wedge}_{\mathrm{SO}(2)} : \mathbb{R}^1 \to \mathfrak{so}(2)$. Its inverse, $(\cdot)^{\vee}_{\mathrm{SO}(2)} : \mathfrak{so}(2) \to \mathbb{R}^1$, follows trivially from relation (3.6). The exponential for $\mathrm{SO}(2)$, performing $\exp_{\mathrm{SO}(2)} : \mathfrak{so}(2) \to \mathrm{SO}(2)$, is given as

$$\exp_{\mathrm{SO}(2)}(x^{\wedge}_{\mathrm{SO}(2)}) = \begin{bmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{bmatrix} , \tag{3.7}$$

while the inverse operator, $\log_{\mathrm{SO}(2)} : \mathrm{SO}(2) \to \mathfrak{so}(2)$, can be evaluated from (3.7). Due to the commutativity of $\mathrm{SO}(2)$, the adjoint operators are given as

$$\mathrm{Ad}_{\mathrm{SO}(2)}(X) = 1 \text{ and } \mathrm{ad}_{\mathrm{SO}(2)}(x) = 0 . \tag{3.8}$$

3.1.3 Special orthogonal group SO(3)

The SO(3) group represents an orientation of a rigid body in 3D space, and is defined as

$$\mathrm{SO}(3) = \left\{ X \in \mathbb{R}^{3\times 3} \mid X^{\mathrm{T}} X = I,\ \det(X) = 1 \right\} . \tag{3.9}$$

For a Euclidean joint space vector $x = [\phi_1\ \phi_2\ \phi_3]^{\mathrm{T}}$, the Lie algebra $\mathfrak{so}(3)$ is given as a skew symmetric matrix

$$x^{\wedge}_{\mathrm{SO}(3)} = \begin{bmatrix} 0 & -\phi_3 & \phi_2 \\ \phi_3 & 0 & -\phi_1 \\ -\phi_2 & \phi_1 & 0 \end{bmatrix} \in \mathfrak{so}(3) . \tag{3.10}$$

where $(\cdot)^{\wedge}_{\mathrm{SO}(3)} : \mathbb{R}^3 \to \mathfrak{so}(3)$. Its inverse, $(\cdot)^{\vee}_{\mathrm{SO}(3)} : \mathfrak{so}(3) \to \mathbb{R}^3$, follows trivially from (3.10). The exponential for SO(3), performing mapping $\exp_{\mathrm{SO}(3)} : \mathfrak{so}(3) \to \mathrm{SO}(3)$, is given as

$$\begin{aligned} \exp_{\mathrm{SO}(3)}(x^{\wedge}_{\mathrm{SO}(3)}) = &\cos(|x|)\mathrm{I}^3 + \\ &+ (1 - \cos(|x|))\frac{xx^{\mathrm{T}}}{|x|^2} + \sin(|x|)\frac{x^{\wedge}_{\mathrm{SO}(3)}}{|x|} . \end{aligned} \tag{3.11}$$

The logarithm, performing mapping $\log_{\mathrm{SO}(3)} : \mathrm{SO}(3) \to \mathfrak{so}(3)$, is given as

$$\begin{aligned} \log_{\mathrm{SO}(3)}(X) = &\frac{\theta}{2\sin(\theta)}(X - X^{\mathrm{T}}) \\ &\text{s.t.} \quad 1 + 2\cos(\theta) = \mathrm{Tr(X)} \\ &\begin{cases} \theta \neq 0 & -\pi < \theta < \pi \\ \theta = 0 & \log(X) = 0 \end{cases} . \end{aligned} \tag{3.12}$$

The adjoints $\mathrm{Ad}_{\mathrm{SO}(3)}$ and $\mathrm{ad}_{\mathrm{SO}(3)}$ are respectively given as

$$\mathrm{Ad}_{\mathrm{SO}(3)}(X) = X \text{ and } \mathrm{ad}_{\mathrm{SO}(3)}(x) = x^{\wedge}_{\mathrm{SO}(3)} . \tag{3.13}$$

3.1.4 Special euclidean group SE(3)

The group SE(3) describes 6 DoF rigid body pose and is formed as a semi-direct product of the Euclidean space vector $\mathbb{R}^3$ and the special orthogonal group SO(3)[1], corresponding to

[1]The euclidean space can be formed only by employing direct product, while other ways to concatenate Lie groups also exist, i.e., semi-direct product, twisted product, etc.

translational and rotational parts, respectively. This group is defined as

$$\mathrm{SE}(3) = \left\{ \begin{pmatrix} R & t \\ \mathbf{0} & 1 \end{pmatrix} \in \mathbb{R}^{4\times 4} \mid \{R, t\} \in \mathrm{SO}(3) \times \mathbb{R}^3 \right\} .$$

For a euclidean space vector representing the pose of a rigid body consisting of a 3DoF position vector t and a 3DoF orientation vector ϕ, where $x = [t\ \phi]^{\mathrm{T}}$, the Lie algebra $\mathfrak{se}(3)$ is

$$x^{\wedge}_{\mathrm{SE}(3)} = \begin{bmatrix} \phi^{\wedge}_{\mathrm{SO}(3)} & t \\ \mathbf{0} & 0 \end{bmatrix} \in \mathfrak{se}(3) . \tag{3.14}$$

where $(\cdot)^{\wedge}_{\mathrm{SE}(3)} : \mathbb{R}^6 \to \mathfrak{se}(3)$. Its inverse, $(\cdot)^{\vee}_{\mathrm{SE}(3)} : \mathfrak{se}(3) \to \mathbb{R}^6$, follows trivially from (3.14). The exponential for $\mathrm{SE}(3)$, performing mapping $\exp_{\mathrm{SE}(3)} : \mathfrak{se}(3) \to \mathrm{SE}(3)$, is given as

$$\begin{aligned} &\exp_{\mathrm{SE}(3)}(x^{\wedge}_{\mathrm{SE}(3)}) = \begin{bmatrix} R & Lt \\ \mathbf{0} & 1 \end{bmatrix} \\ &R = \exp_{\mathrm{SO}(3)}(\phi^{\wedge}_{\mathrm{SO}(3)}) \\ &L = \frac{\sin(|\phi|)}{|\phi|}\mathrm{I}^3 + (1 - \frac{\sin(|\phi|)}{|\phi|})\frac{\phi\phi^{\mathrm{T}}}{|\phi|^2} + \frac{1-\cos(|\phi|)}{|\phi|^2}\phi^{\wedge}_{\mathrm{SO}(3)}. \end{aligned} \tag{3.15}$$

The logarithm, performing mapping $\log_{\mathrm{SE}(3)} : \mathrm{SE}(3) \to \mathfrak{se}(3)$, is calculated by deconstructing X, and determining ϕ by using (3.12). Then, from (3.15) we can determine t.

In order to determine the adjoints for $\mathrm{SE}(3)$, we need to deconstruct the state $X \in \mathrm{SE}(3)$ and vector $x \in \mathbb{R}^6$. Firstly, we extract the rotation part C and translation part t from X, and secondly, we split the translation part t and orientation part ϕ from x. Then, the adjoints $\mathrm{Ad}_{\mathrm{SE}(3)}$ and $\mathrm{ad}_{\mathrm{SE}(3)}$ are

$$\mathrm{Ad}_{\mathrm{SE}(3)}(X) = \begin{bmatrix} R & tR \\ \mathbf{0} & R \end{bmatrix}, \ \mathrm{ad}_{\mathrm{SE}(3)}(x) = \begin{bmatrix} \phi^{\wedge}_{\mathrm{SO}(3)} & t^{\wedge}_{\mathrm{SO}(3)} \\ \mathbf{0} & \phi^{\wedge}_{\mathrm{SO}(3)} \end{bmatrix} .$$

3.2 Kalman Filter

The Kalman Filter [109] is a popular sensor fusion technique which provides an unbiased estimate the state of a system from noisy observations. For a linear model it is shown to be an optimal filter under the assumption that both measurement and process noise are zero-mean Gaussian.

Consider a linear time-invariant system of the following form:

$$x_t = Ax_{t-1} + w_{t-1} \tag{3.16}$$

$$z_t = Cx_t + v_t \tag{3.17}$$

At each time step t, x_t is the n-dimensional state vector which represents the true state of the system, the matrix A is the process matrix describing the transition from the previous state x_{t-1} to the current state x_t. The process update is assumed to be affected by zero mean Gaussian noise w_t with covariance Q_t. The measurement vector made at each time step is z_t and the matrix C is a transformation from state space into measurement space. Any noise in the measurement v_t is modeled as zero mean Gaussian with covariance R_t. The goal is to make an unbiased estimate of the state $\hat{x}_t$ at every time step to minimize the error covariance matrix P_t.

$$P_t = E[(x_t - \hat{x}_t)(x_t - \hat{x}_t)^T] \tag{3.18}$$

For a linear system with Gaussian process and measurement noise, the Kalman filter finds the optimal gain K_t used to combine the priori state estimate $\hat{x}_t^-$ made using equation (3.16) and the current measurement z_t for optimal state estimate $\hat{x}_t$ in a closed form. The state estimate is then calculated as

$$\hat{x} = \hat{x}_t^- + K_t(z_t - C\hat{x}_t^-) \tag{3.19}$$

where $z_t - C\hat{x}_t^-$ is the residual error between the actual and predicted measurement. Kalman filter equations can be separated into two steps, prediction and update. During the prediction step, the priori state and error covariance estimates are made. These estimates are then modified in the update step using the measurement. Figure 3.2 summarizes the two steps.

state prediction equations	measurement update equations
$\hat{x}_t^- = A\hat{x}_{t-1}$	$K_t = P_t^- C^T (CP_t^- C^T + R_t)^{-1}$
$P_t^- = AP_{t-1}A^T + Q_t$	$\hat{x}_t = \hat{x}_t^- + K_t(z_t - C\hat{x}_t^-)$
	$P_t = (I - K_t C)P_t^-$

Figure 3.2 Kalman filter state prediction and update.

3.2.1 Extended Kalman Filter

Many real world systems are non-linear and have the form

$$x_t = f(x_{t-1}) + w_{t-1} \tag{3.20}$$

$$z_t = h(x_t) + v_t. \tag{3.21}$$

thus the Kalman filter cannot be applied directly. The Extended Kalman Filter (EKF) linearizes the equations about the operating point, approximating the system as

$$x_t \approx \tilde{x}_t + A_t(x_t - \tilde{x}_t) + w_{t-1} \tag{3.22}$$

$$z_t \approx \tilde{z}_t + C_t(x_t - \tilde{x}_t) + v_t \tag{3.23}$$

where $A_t = \frac{\partial f}{\partial x_t}$ and $C_t = \frac{\partial h}{\partial x_t}$ are the Jacobians of the state update and measurement equations with respect to the state, $\tilde{x} = f(x_{t-1}, 0)$ and $\tilde{z} = h(x_t, 0)$, the noiseless state and measurement estimates. The Kalman gain is computed but may no longer be optimal due loss of higher order terms during linearization. The equations are again separated into prediction and update steps and are summarized in figure 3.2.

state prediction	measurement update
$\hat{x}_t^- = f(x_{t-1}, w = 0)$	$P_y = C_t P_t^- C_t^T + V_t R_t V_t^T$
$P_t^- = A_t P_{t-1} A_t^T + W_t Q_{t-1} W_t^T$	$K_t = P_t^- C_t^T (P_y)^{-1}$
	$\hat{x}_t = \hat{x}_t^- + K_t(z_t - h(\hat{x}_t^-, 0))$
	$P_t = (I - K_t C_t) P_t^-$

Figure 3.3 Extended Kalman filter state prediction and update.

3.2.2 Unscented Kalman Filter

A key part of the Kalman filter is the propagation of uncertainty in the state represented by a multivariate Gaussian through the system dynamics. When the system dynamics are non-linear the extended Kalman filter analytically propagates this uncertainty using first order linearization. If the first order linearization does not closely match the dynamics this can introduce large errors in the posterior state estimate. The unscented Kalman filter instead is a sampling method that carefully choses state samples and propagates them through the full non-linear dynamics of the system. After the propagation the mean and covariance of the posterior are re-computed. This approach accurately captures the posterior to the third order of Taylor series expansion.

Consider again the stochastic representation of the state variable at time t with mean $\hat{x}_t$ and covariance P_t. The unscented Kalman filter samples the state variable and propagates the samples through the true non-linear dynamic system described by equation (3.20). The state samples are chosen in accordance with the unscented transform which is a technique for calculating statistics

of a normal random variable undergoing non-linear transformation [110]. The propagated state samples are chosen and weighted as follows:

$$ {}^{0}x_t = \hat{x}_t \tag{3.24} $$

$$ {}^{i}x_t = \hat{x}_t + ((n+\lambda)P_t)_i^{\frac{1}{2}}, \quad i = 1 \dots n \tag{3.25} $$

$$ {}^{i}x_t = \hat{x}_t - ((n+\lambda)P_t)_{i-n}^{\frac{1}{2}}, \quad i = n+1 \dots 2n \tag{3.26} $$

$$ {}^{0}W_\mu = \frac{\lambda}{n+\lambda} \tag{3.27} $$

$$ {}^{0}W_P = \frac{\lambda}{n+\lambda} + (1 - \alpha^2 + \beta) \tag{3.28} $$

$$ {}^{i}W_P = {}^{i}W_\mu = \frac{1}{2(n+\lambda)}, \qquad i = 1 \dots 2n \tag{3.29} $$

where n is the size of the state vector, $\lambda = \alpha^2 n - n$ is a scaling factor, and α controls how far the chosen samples are spread from the mean. $((n+\lambda)P_t)_i^{\frac{1}{2}}$ denotes the i_{th} column of the matrix square root. The propagated state mean and covariance are then computed as a weighted sum of the propagated samples. The propagated samples are also used to calculate the measurement prediction using the non-linear measurement equation (3.21). Unscented Kalman filter equations are summarized in Figure 3.4.

state prediction	measurement update
$\hat{x}_t^- = \sum_0^{2n} {}^{i}W_\mu f({}^{i}x_t)$	$P_{zz} = \sum_0^{2n} {}^{i}W_P (h(f({}^{i}x_t)) - \hat{z}_t^-)(h(f({}^{i}x_t)) - \hat{z}_t^-)^T$
$P_t^- = \sum_0^{2L} {}^{i}W_P (f({}^{i}x_t) - \hat{x}_t^-)(f({}^{i}x_t) - \hat{x}_t^-)^T$	$P_{xz} = \sum_0^{2n} {}^{i}W_P (f({}^{i}x_t) - \hat{x}_t^-)(h(f({}^{i}x_t)) - \hat{z}_t^-)^T$
$\hat{z}_t^- = \sum_0^{2n} {}^{i}W_\mu h(f({}^{i}x_t))$	$K_t = P_{xz} P_{zz}^{-1}$
	$\hat{x}_t = \hat{x}_t^- + K_t(z_t - \hat{z}_t^-)$
	$P_t = P_t^- - K_t P_{zz} K_t^T$

Figure 3.4 Unscented Kalman filter state prediction and update.

While UKF does correctly estimate the mean and covariance up to third order it requires propagation of the $2n$ selected state points through the non-linear dynamics and measurement equations. This makes UKF significantly more computationally expensive than EKF and for large state vectors it quickly becomes intractable to utilize the algorithm for real time state estimation.

3.3 Gaussian Processes

Gaussian Processes (GPs) are a non-parametric function and covariance approximation method. Formally, a GP is defined as "collection of random variables any finite number of which have a joint Gaussian distribution" [111]. They have excellent regression capabilities, providing a non-parametric, highly non-linear model. Furthermore, due to their probabilistic nature, GPs allow estimating the uncertainty at the output. They have been used extensively in many applications, including geostatistics [112], robotic modeling and control [113, 114], and finance [115]. However, the computational complexity of both learning a GP model and utilizing it for regression grows exponentially with the number of training data samples. This has limited their application to smaller data sets.

A GP $f(x) \sim GP(m(x), k(x, x'))$ is completely specified by its mean and covariance functions, $m(x)$ and $k(x, x')$ respectively.

$$m(x) = \mathbb{E}(f(x)) \tag{3.30}$$

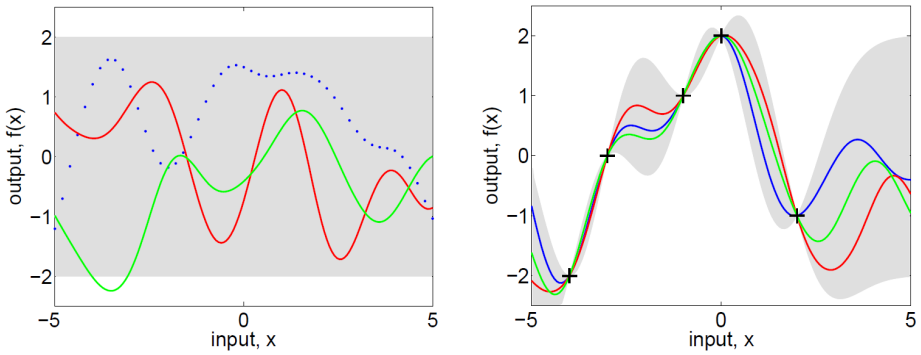
$$k(x, x^{'}) = \mathbb{E}\big((f(x) - m(x))(f(x') - m(x'))^T\big) \tag{3.31}$$

Notice that the covariance of $f(x)$ is dependent only on the input x and is calculated using a kernel function $k(x, x')$. A kernel is any function that is symmetric and positive definite, leading to a valid positive symmetric definite GP covariance for any input x. We write the GP as $f(x) \sim GP(m(x), k(x, x'))$, the random variables are thus the value of f at location x.

Consider a set of N observations experiencing zero mean Gaussian noise $(x_i, y_i)|i \in 1, 2 \ldots N$ where $y_i = f(x_i) + \epsilon$, with $\epsilon \sim \mathcal{N}(0, \sigma_n^2)$. Assuming a zero mean function $\mu(x) = 0$ and since the noise is independent, $y \sim \mathcal{N}(0, K_{xx} + \Sigma_N)$ where $y = [y_1, y_2 \ldots y_N]^T$, $x = [x_1, x_2 \ldots x_N]^T$, K_{xx} is the kernel matrix $k(x, x)$, and $\Sigma_N = \sigma_n^2 I_N$, I_N being an $N \times N$ identity matrix. Consider now m previously unseen test points $x_* = [x_1, x_2 \ldots x_m]^T$, the joint distribution of the noise free training data and the previously unseen test points can be written as:

$$\begin{bmatrix} f \\ f_* \end{bmatrix} \sim \mathcal{N}\Big(0, \begin{bmatrix} k(x, x) & k(x, x_*) \\ k(x_*, x) & k(x_*, x_*) \end{bmatrix}\Big). \tag{3.32}$$

Incorporating the independent zero mean Gaussian noise ϵ at each output sample and using the fact that the mean and covariance of jointly Gaussian random variables are also Gaussian, the



(a), prior (b), posterior

Figure 3.5 Left (a) shows three random functions drawn from GP prior before conditioning on the training data. Right (b) shows three random functions drawn from the GP after conditioning on 5 training samples. The shaded area represents the 95% confidence region. The figure is taken from [116].

mean and covariance of the unseen test points is computed as:

$$y_*|x, y, x_* \sim \mathcal{N}(\mu_*, \Sigma_*) \tag{3.33}$$

$$\begin{aligned}\mu_* &= \mathbb{E}(y_*|x, y, x_*) \\ &= K_{x_*x}(K_{xx} + \Sigma_N)^{-1} y\end{aligned} \tag{3.34}$$

$$\Sigma_* = \underbrace{K_{x_*x_*}}_{m\times m} - \underbrace{K_{x_*x}}_{m\times N} \underbrace{(K_{xx} + \Sigma_N)^{-1}}_{N\times N} \underbrace{K_{xx_*}}_{N\times m} \tag{3.35}$$

Figure 3.5 shows three random functions drawn from a GP prior and posterior after conditioning on 5 observations. The size of the training datset N and the size of the test point vector m determine the computational requirements during inference. While the $N \times N$ matrix $(K_{xx} + \Sigma_N)^{-1}$ is constant given training data, the $m \times N$ matrix K_{x_*x} must be computed and multiplied with the $N \times N$ matrix to predict m points.

3.3.1 Hyperparameter Optimization

Typically the chosen kernel function K_{xx} will have multiple tuning hyperparameters θ. A common way to find the optimal parameters θ_* for the given training data is to maximize the marginal likelihood. Consider the probability of seeing the training output given the training input assuming that the data belongs to a zero mean multivariate normal distribution

$$P(y|x) = (2\pi)^{\frac{-N}{2}} |K_{xx} + \Sigma_N|^{\frac{-1}{2}} e^{-\frac{1}{2} y^T (K_{xx}+\Sigma_N)^{-1} y} \tag{3.36}$$

taking the log of the probability changes the equation into a form easily differentiable with respect to K_{xx} and thus also the tuning hyperparameters

$$log(P(y|x)) = -\frac{1}{2}y^T(K_{xx} + \Sigma_N)^{-1}y - \frac{1}{2}log(|K_{xx} + \Sigma_N|) - \frac{N}{2}log(2\pi) \tag{3.37}$$

Letting $K = K_{xx}+\Sigma_N$ and differentiating the log likelihood with respect to the j_{th} tuning parameter, gradient descent methods can be utilized to iteratively find θ_*.

$$\frac{\partial log(P(y|x))}{\partial \theta_j} = \frac{1}{2}y^T K^{-1}\frac{\partial K}{\partial \theta_j}K^{-1}y - \frac{1}{2}tr(K^{-1}\frac{\partial K}{\partial \theta_j}) \tag{3.38}$$

To achieve the form above the following matrix identity

$$\frac{\partial M^{-1}(\theta)}{\partial \theta} = -M^{-1}(\theta)\frac{\partial M(\theta)}{\partial \theta}M^{-1}(\theta) \tag{3.39}$$

and the Jacobi's formula

$$\frac{\partial |M(\theta)|}{\partial \theta} = tr(adj(M(\theta))\frac{\partial M(\theta)}{\partial \theta}) \tag{3.40}$$

are used. Where $M(\theta)$ is any matrix parameterized by θ.

Notice that at each iteration of gradient descent the $N \times N$ matrix K needs to be computed and then inverted leading to $\mathcal{O}(N^3)$ operations. This makes GPs limited to smaller training datasets.

3.3.2 Computational Complexity

We see that Gaussian process computational complexity for both training and regression is determined by the number of the training points. For a long time this has limited the application of GPs to small datasets. One way to reduce the computational complexity is to assume that the entire dataset contains redundant information and thus the GP can be accurately approximated by choosing a smaller set of h inducing points. This can be viewed as approximating the full kernel covariance matrix with one of lower rank [117]. Multiple inducing point approximations have been proposed [118–120]. Minimizing the Kullback Leibler divergence between the approximate and full posterior processes allows to optimize both the selection of the inducing points and the kernel hyperparameters [121]. Selecting different inducing points for the mean and covariance estimation (decoupling basis) allows to model more complex mean functions while maintaining computationally tractable covariance [122]. Setting the basis of the mean to contain the basis of the covariance and an additional orthogonal component ensures that the components can be

optimized separately [123]. Similarly, the full GP can be thought of as combination of two independent processes with inducing points, the variation not captured by the inducing points of the first is contained in the second [124].

Instead of choosing the inducing points, other approaches to reduce computational complexity focus on approximations of the kernel function. It is possible to approximate certain kernels as an output from a linear, time invariant, stochastic system of finite order [125]. In this form, the required numerical problems deal with symmetric block-tridiagonal matrices and can utilize parallelization to further speed up computation [126]. Combining both the inducing points and state space approximations leads to "double sparse" GPs, further decreasing the complexity and storage requirements [127]. Similar to the state space approximation one can approximate any kernel as a finite Fourier series and optimize over both the selected frequencies and their coefficients [128]. Finally, approaches approximating the Kernel matrix using eigen values and vectors have been proposed. Williams and Seeger [129] used the Nystrom method to approximate the first n eigenvalues and vectors from the training data. Peng and Qi [130] expanded the approach to maximize the marginal likelihood and include hyperparameter training.

Most of the aforementioned works tackle GP's computational complexity with the goal of training on extremely large data sets and consider the faster regressions as an added benefit. However, recently GPs are finding novel control applications such as robots learning from human demonstrations [131], estimating manipulator dynamics [132], and controlling a drone [133]. When GPs are used in such online control scenarios the regression computational efficiency is of paramount importance since the control loop often must run at very high frequency in order for the system to quickly respond to disturbances. Training on the other hand can be done offline with previously collected data.

3.3.3 Kernel Functions

The kernel function $K_{xx'}$ is a crucial component of the Gaussian processes since it encodes the similarity between near data points and enforces assumptions about the function we aim to learn. It is important to note that a valid kernel function must be continuous, differential, symmetric, and positive definite. These properties ensure that a valid output covariance matrix is generated by the kernel function required for GP. It also leads to the nice property that multiple of two or more kernel functions produces a valid kernel as well, allowing to capture complex trends in the data using combinations of simple kernels. This section reviews the three most popular kernel functions, we exclude the commonly placed scaling parameter in front of each kernel since it is redundant in the formulation of multi-output Gaussian processes explained in the next section.

3.3.3.1 Linear Kernel

The linear kernel is the inner product of x and x'.

$$k_{lin}(x, x') = x^T x' \tag{3.41}$$

Unlike the other two kernels presented in this section it is non-stationary and thus does not only depend on the relative position of the two input vectors. Utilizing only the linear kernel in a GP is equivalent to Bayesian linear regression for which there are much more efficient methods than inverting the $N \times N$ kernel matrix. Thus the linear kernel is typically seen in combination with others when some linear relationship in the data is noted.

3.3.3.2 Squared Exponential Kernel

The squared exponential is the most popular kernel function utilized in Gaussian processes and support vector machines.

$$k_{se}(x, x') = e^{\frac{-\|x-x'\|^2}{2l_{se}^2}} \tag{3.42}$$

It has a single hyperparameter l_{se} which controls the kernel width. It is a stationary and infinitely differentiable kernel. This makes it particularly appealing for Gaussian processes and it is typically used as the default initial kernel when training on a new data-set.

3.3.3.3 Periodic Kernel

Incorporating a sinusoid into the squared exponential kernel produces the periodic covariance function

$$k_{pr}(x, x) = e^{-\frac{2sin(f_{pr}\frac{\|x-x'\|}{2})^2}{w_{pr}^2}} \tag{3.43}$$

which allows to create Gaussian processes that are repeat themselves at a certain frequency. The frequency parameter f_{pr} determines the distance between the repetitions and the width w_{pr} controls the kernel width i.e. the similarity between near data points.

3.3.4 Multi-output Gaussian Processes

A simple way to handle multioutput modelling using GPs is to assume that the outputs are independent and train a separate GP for each. However, this approach cannot capture the correlation between different outputs present in the training data. By vectorizing the multioutput training data it is possible to capture cross output correlation [134]. Consider learning a GP representation of a function with M outputs, provided the training pairs x_i, $[y_i^1 \dots y_i^M]$, re-define the training data as $y = [y_1^1\ y_2^1 \dots y_N^1\ y_1^2 \dots y_N^2 \dots y_N^M]^T$, vectorizing all of the outputs. We now consider the $NM \times NM$ covariance matrix of y

$$K = K_f \otimes K_{xx} + \Sigma_{NM} \tag{3.44}$$

where K_f is an $M \times M$ positive symmetric definite matrix that describes output similarities and provides scaling to the kernel matrix K_{xx}. Σ_{NM} is an $NM \times NM$ matrix describing the training observation noise that now may include covariance between outputs. Finally, $\otimes$ denotes the Kronecker product. Note that setting K_f to the identity matrix and keeping Σ_{NM} diagonal implies independent outputs similar to training a separate GP for each. Inference can be done for multiple outputs by substitution $K_f \otimes K_{x_*x}$ for K_{x_*x}. Note that to guarantee that the output covariance matrix remains symmetric positive definite during training and prediction K_f must also be symmetric positive definite.

Now, in addition to the kernel function hyperparameters we would like to learn the output cross correlation matrix K_f from the training data by including it in the marginal log likelihood maximization. Parameterizing $K_f = LL^T$ using Cholesky decomposition allows to apply regular gradient descent maximizing marginal log likelihood in terms of L and ensuring that K_f remains symmetric positive definite

$$\frac{\partial log(P(y|x))}{\partial L} = \frac{1}{2} y^T K^{-1} \frac{\partial K}{\partial L} K^{-1} y - \frac{1}{2} tr(K^{-1} \frac{\partial K}{\partial L}) \tag{3.45}$$

where $\frac{\partial K}{\partial L} = \frac{\partial LL^T}{\partial L} \otimes K_{xx} + \Sigma_{NM}$ and $\frac{\partial LL^T}{\partial L}$ can be efficiently calculated as $\frac{\partial LL^T}{\partial L} = (I_{(MM)^2} + T) I_{MM} \otimes L$ where T is a transformation matrix such that $Tvec(L) = vec(L^T)$ [135]. Since valid L is lower triangular only those entries are updated during gradient descent iterations.

3.4 Model Predictive Control

Model predictive control uses the dynamical model of the plant to predict the state x_t over a horizon of length N and finds the optimal control law u_k to minimize a cost function. When

dealing with linear systems of the form

$$x_{t+1} = Ax_t + Bu_t \tag{3.46}$$

and a quadratic cost function

$$J = \sum_{i=t+1}^{i=t+N} x_i^T Q_i x_i + u_i^T R_i u_i \tag{3.47}$$

where Q_i and R_i are positive definite matrices, the optimal control signal can be efficiently computed while satisfying linear constraints on the state and control signal.

$$u_{min} \leq P_u u_t \leq u_{max} \tag{3.48}$$

$$x_{min} \leq P_x x_t \leq x_{max} \tag{3.49}$$

By re-writing the state equations over a predictive horizon of length N only in terms of the control input $u_{1:N}$:

$$\begin{bmatrix} x_{t+1} \\ x_{t+2} \\ \vdots \\ x_{t+N} \end{bmatrix} = \underbrace{\begin{bmatrix} B & 0 & \cdots & \\ AB & B & 0 \cdots & \\ \vdots & & & \\ A^{N-1}B & A^{N-2}B & \cdots & B \end{bmatrix}}_{\bar{S}} \underbrace{\begin{bmatrix} u_t \\ u_{t+1} \\ \vdots \\ u_{t+N-1} \end{bmatrix}}_{u_{1:N}} + \underbrace{\begin{bmatrix} A \\ A^2 \\ \vdots \\ A^N \end{bmatrix}}_{\bar{T}} x_t \tag{3.50}$$

the cost function appears in quadratic form

$$J = u_{1:N}^T \underbrace{(\bar{R} + \bar{S}^T \bar{Q} \bar{S})}_{H} u_{1:N} + \underbrace{x_k 2\bar{T}^T \bar{Q} \bar{S}}_{f^T} u_{1:N}. \tag{3.51}$$

where $\bar{Q}$ and $\bar{R}$ are block diagonal concatenations of Q_i and R_i over the horizon. Standard quadratic programming methods can then be used to solve for the optimal $u_{1:N}$. The first optimal control signal is applied to the plant and the process is repeated [136].

Chapter 4

Human motion estimation on Lie groups

The work presented in this chapter was done in collaboration with Josip Cesic who at the time was a PhD student ot University of Zagreb. A version of this chapter was published in IEEE Transactions on Cybernetics [137]

In this chapter we present an algorithm for full body human motion estimation on Lie groups, which uses either 3D marker position measurements and/or IMU measurements. The proposed filtering approach performs stochastic inference of human motion by defining the state space to reside on a Lie group, with each state element corresponding to the kinematic model of an analysed human body part. The geometry of the state space is explicitly taken into account by applying Lie group Extended Kalman Filter (LG-EKF), where the prediction step is assumed to follow a constant acceleration model. We provide an extensive experimental evaluation of the proposed estimation approach in both simulations and by using new real-world datasets. Finally, we consider the problem of observability of a human body by relying on the 3D marker position measurements. For this purpose we use an approach relying on evaluation of Lie derivatives, and analyze the observability of a chain of $SO(3)$ elements, which can be seen as a generalization of human body kinematics. The results of the observability analyisis are validated by emulating kinematics of a human arm.

4.1 Motion estimation on Lie groups

4.1.1 Construction of the state space

We construct the state space by using Lie group representatives for each joint of interest. One dof revolute joints are represented with the special orthogonal group SO(2), while 3 dof spherical joints are modelled with the special orthogonal group SO(3). To localize the human in an inertial frame, we use a special euclidean group member SE(3) for connecting the origin of the frame with the base of the body, modeling both translational and rotational motion. Finally, the state of the system is constructed by concatenating Lie group members via a Cartesian product, starting with SE(3), and extending with either SO(2) or SO(3) groups.

As an example we consider a state space model of the full human body employing either Lie groups or Euler angles, as illustrated in Fig. 4.1. An example of the group representing the positional variables for this model is

$$\begin{aligned} \mathrm{G}_{\mathrm{pos}} = \overbrace{\mathrm{SE}(3)}^{\text{base pose}} \times \overbrace{\mathrm{SO}(3)}^{\text{L hip}} \times \overbrace{\mathrm{SO}(3)}^{\text{R hip}} \times \overbrace{\mathrm{SO}(2)}^{\text{L knee}} \times \overbrace{\mathrm{SO}(2)}^{\text{R knee}} \times \\ \times \overbrace{\mathrm{SO}(2)^2}^{\text{L ankle}} \times \overbrace{\mathrm{SO}(2)^2}^{\text{R ankle}} \times \overbrace{\mathrm{SO}(2)}^{\text{L toes}} \times \overbrace{\mathrm{SO}(2)}^{\text{R toes}} \times \overbrace{\mathrm{SO}(3)}^{\text{neck}} \times \\ \times \overbrace{\mathrm{SO}(3)}^{\text{L shoulder}} \times \overbrace{\mathrm{SO}(3)}^{\text{R shoulder}} \times \overbrace{\mathrm{SO}(2)^2}^{\text{L elbow}} \times \overbrace{\mathrm{SO}(2)^2}^{\text{R elbow}}, \end{aligned} \tag{4.1}$$

where $\mathrm{SO}(2)^2 = \mathrm{SO}(2) \times \mathrm{SO}(2)$. Alongside positional variables, in this work we also want to estimate joint velocities and accelerations. For this purpose we associate velocity and acceleration to each dof of each joint of the body, where each component is represented as a real-numbered value. The full state of the system X_k at time instant k is then of the form

$$\begin{aligned} X_k &= \mathrm{blkdiag}\{\theta_k, \omega_k, \alpha_k\} \in \mathrm{G} \\ \theta_k &= \mathrm{blkdiag}\{\theta_k^1, ..., \theta_k^n\} \in \mathrm{G}_{\mathrm{pos}} \\ \omega_k &= \mathrm{blkdiag}\{\omega_k^1, ..., \omega_k^n\} \in \mathbb{R}^{p_1} \times \cdots \times \mathbb{R}^{p_n} \\ \alpha_k &= \mathrm{blkdiag}\{\alpha_k^1, ..., \alpha_k^n\} \in \mathbb{R}^{p_1} \times \cdots \times \mathbb{R}^{p_n}, \end{aligned} \tag{4.2}$$

where θ_k^i is the position of the i-th joint, ω_k^i is the velocity of the i-th joint[1], $\alpha_k^i \in \mathbb{R}^{p_i}$ is the acceleration of the i-th joint, n is the number of joints of a body, while p_i is the number of dofs of the i-th joint. blkdiag refers to stacking matrices diagonally and filling all other entries with zeros.

[1]Euclidean space $\mathbb{R}_i^p$, $p \in \mathbb{N}$ is also Lie group and in order to construct G we employ its matrix representation obtained by simple matrix embedding. The matrix representation of the Euclidean space is also a subgroup of SE(n) where a pure translation is employed [108].

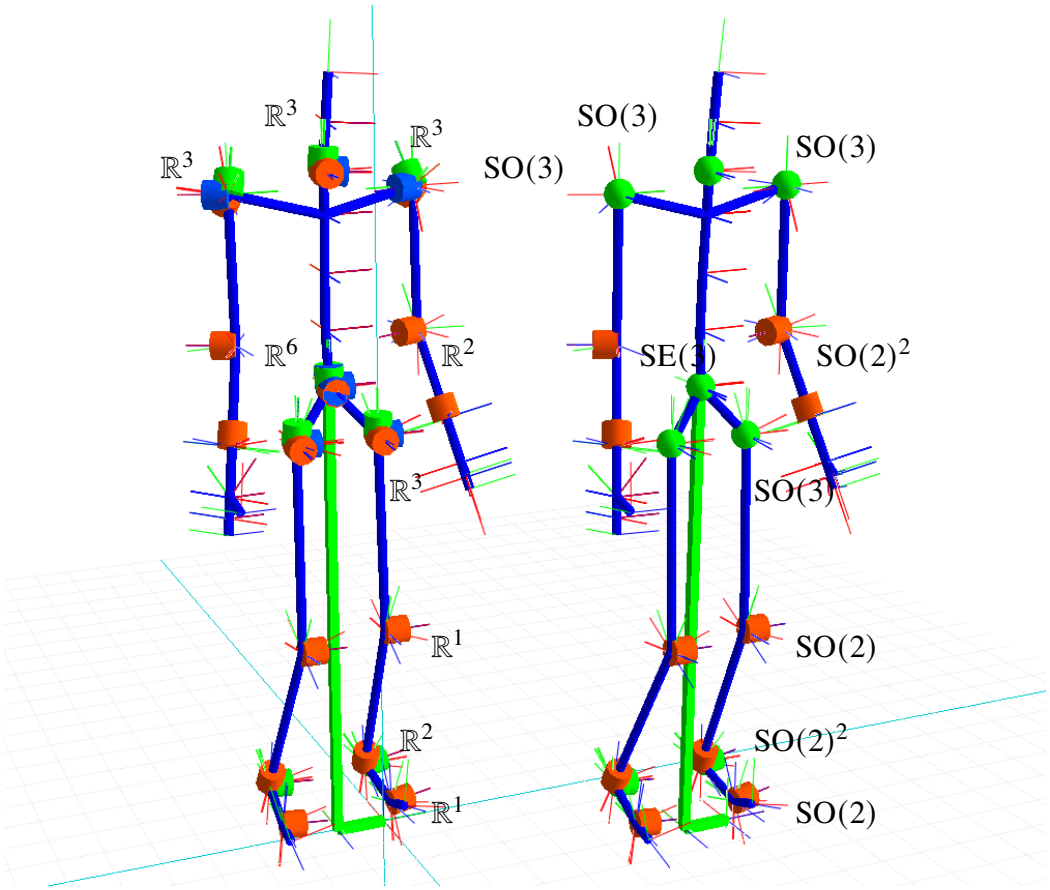


Figure 4.1 Comparison of Euler angle (left) and Lie group (right) skeleton models. Revolute SO(2) joints are red cylinders, spherical SO(3) joints are green spheres, prismatic SE(3) joints are green square rods.

4.1.2 Motion prediction step

We assume that the motion model of the system can be described with the following equation [62]

$$X_{k+1} = f(X_k, n_k) = X_k \, \exp_{\mathrm{G}}^{\wedge}\left(\hat{\Omega}_k + n_k\right), \tag{4.3}$$

where $X_k \in \mathrm{G}$ is the state of the system at time k (4.2), $n_k \sim \mathcal{N}_{\mathbb{R}^p}(\mathbf{0}^{p\times 1}, Q_k)$ is zero mean white Gaussian noise with covariance Q_k, and $\hat{\Omega}_k = \Omega(X_k) : \mathrm{G} \to \mathbb{R}^p$ is a non-linear $\mathcal{C}^2$ function.

In this work we assume the human motion follows a constant acceleration model, hence the

motion model of a single joint i is given as

$$\hat{\Omega}_k^i = \begin{bmatrix} \tau\omega_k^i + \frac{\tau^2}{2}\alpha_k^i \\ \tau\alpha_k^i \\ 0 \end{bmatrix} \in \mathbb{R}^{p_i}, n_k = \begin{bmatrix} \frac{\tau^2}{2}n_k^i \\ \tau n_k^i \\ n_k^i \end{bmatrix} \in \mathbb{R}_i^p, \tag{4.4}$$

where τ is the sampling period and the term n_k^i represents the acceleration increment during the k-th sample [138].

Assuming that the posterior distribution at step k follows the CGD $\mathcal{G}(\mu_k, P_k)$, the resulting prediction can be approximated with a CGD $\mathcal{G}(\mu_{k+1|k}, P_{k+1|k})$. The mean is propagated by

$$\mu_{k+1|k} = \mu_k \exp_{\mathrm{G}}^{\wedge}\left(\hat{\Omega}_k\right), \tag{4.5}$$

while the covariance prediction is computed as

$$P_{k+1|k} = \mathcal{F}_k P_k \mathcal{F}_k^{\mathrm{T}} + \Phi_{\mathrm{G}}(\hat{\Omega}_k) Q_k \Phi_{\mathrm{G}}(\hat{\Omega}_k)^{\mathrm{T}}. \tag{4.6}$$

The operator $\mathcal{F}_k$ can be seen as the matrix Lie group equivalent to the Jacobian of $f(X_k, n_k)$, and is calculated by

$$\begin{aligned} \mathcal{F}_k &= \mathrm{Ad}_{\mathrm{G}}\left(\exp_{\mathrm{G}}^{\wedge}\left(-\hat{\Omega}_k\right)\right) + \Phi_{\mathrm{G}}(\hat{\Omega}_k)\mathscr{L}_k \\ \mathscr{L}_k &= \frac{\partial}{\partial \epsilon}\Omega\left(\mu_k \exp_{\mathrm{G}}^{\wedge}(\epsilon)\right)_{|\epsilon=0}. \end{aligned} \tag{4.7}$$

The term $\mathscr{L}_k$ represents the linearisation term where the argument of the motion model is the mean of the current state X_k with an incremental perturbation added in each of the p directions. Contrary to the conventional EKF, linear additive process noise affects the system as a function of the current state of the system over the transformation $\Phi_{\mathrm{G}}(\hat{\Omega}_k) Q_k \Phi_{\mathrm{G}}(\hat{\Omega}_k)^{\mathrm{T}}$, where Φ_{G} appears due to the displacement of the tangential space during the prediction step, and is given by

$$\Phi_{\mathrm{G}}(v) = \sum_{i=0}^{\infty} \frac{(-1)^i}{(i+1)!} \mathrm{ad}_{\mathrm{G}}(v)^i, \; v \in \mathbb{R}^p. \tag{4.8}$$

Note that here we use the discrete LG-EKF, which has been extended to continuous discrete version in [62]. However, even though continuous prediction can potentially improve the accuracy of the motion prediction step, in our case for the full body skeleton the required numerical integration would induce higher computation costs and it would not be possible to implement the approach for a real time motion capture system.

4.1.3 Measurement update step

Next derive the update step based on the sensors attached to a human body. The discrete measurement model on the matrix Lie group is given as

$$Z_{k+1} = h(X_{k+1}) \exp_{\mathrm{G}'}^{\wedge}(m_{k+1}) , \tag{4.9}$$

where $Z_{k+1} \in \mathrm{G}'$, $h : \mathrm{G} \to \mathrm{G}'$ is a $\mathcal{C}^1$ function, G' is a p'-dimensional Lie group and $m_{k+1} \sim \mathcal{N}_{\mathbb{R}^q}(\mathbf{0}^{q\times 1}, R_{k+1})$ is zero-mean white Gaussian noise with covariance R_{k+1}. The update step of the filter strongly resembles the standard EKF update procedure, relying on the Kalman gain K_{k+1} and innovation vector ν_{k+1} calculated as

$$\begin{aligned} K_{k+1} &= P_{k+1|k}\mathcal{H}_{k+1}^{\mathrm{T}}\left(\mathcal{H}_{k+1}P_{k+1|k}\mathcal{H}_{k+1}^{\mathrm{T}} + R_{k+1}\right)^{-1} \\ \nu_{k+1} &= K_{k+1}\log_{\mathrm{G}'}^{\vee}\left(h(\mu_{k+1|k})^{-1}Z_{k+1}\right) . \end{aligned} \tag{4.10}$$

The matrix $\mathcal{H}_{k+1}$ can be seen as the matrix Lie group equivalent of the Jacobian of $h(X_{k+1})$, and is given as

$$\mathcal{H}_{k+1} = \frac{\partial}{\partial \epsilon}\log_{\mathrm{G}'}^{\vee}\left(h(\mu_{k+1|k})^{-1}h(\mu_{k+1|k}^{\epsilon})\right)_{|\epsilon=0} , \tag{4.11}$$

where $h(\mu_{k+1|k}^{\epsilon}) = h(\mu_{k+1|k}\exp_{\mathrm{G}}^{\wedge}(\epsilon))$, describes the variation of measurements for an infinitesimal motion ϵ. Finally, the measurement update step is calculated as

$$\begin{aligned} \mu_{k+1} &= \mu_{k+1|k}\exp_{\mathrm{G}}^{\wedge}\left(\nu_{k+1}\right) \\ P_{k+1} &= \Phi_{\mathrm{G}}\left(\nu_{k+1}\right)\left(I - K_{k+1}\mathcal{H}_{k+1}\right)P_{k+1|k}\Phi_{\mathrm{G}}\left(\nu_{k+1}\right)^{\mathrm{T}} . \end{aligned} \tag{4.12}$$

For a more formal derivation, the reader is referred to [62]. We now evaluate the matrix $\mathcal{H}_{k+1}$ based on marker, gyro and accelerometer measurements.

4.1.4 Marker update

The marker measurement function is given as

$$h_m(X_{k+1|k}) = \mathcal{K}_s^0\vec{o} , \tag{4.13}$$

where the term $\mathcal{K}_s^0 = \mathcal{K}_s^0(X_{k+1|k}) \in \mathrm{SE}(3)$ represents the forward kinematics of the position of marker s, and $\vec{o} \in \mathbb{R}^4$ is the origin represented in homogeneous coordinates.

Substituting the measurement model for the marker into the Lie group Jacobian 4.11. The part of $\mathcal{H}_{k+1}$ corresponding to a marker measurement with respect to joint orientation θ evaluates as follows

$$
\begin{aligned}
{}^{m}\mathcal{H}^{\theta}_{k+1} =& \frac{\partial}{\partial \epsilon}\left(\log_{\mathrm{G}'}\left(h\left(\mathcal{K}^0_s(X_{k+1|k})\right)^{-1}\ h\left(\mathcal{K}^0_s(X^{\epsilon}_{k+1|k})\right)\right)\right)^{\vee}_{\mathrm{G}'\ |\epsilon=0} \\
=& \frac{\partial}{\partial \epsilon}\left(\log_{\mathrm{G}'}\left(\begin{bmatrix} I & \mathcal{K}^0_s(X^{\epsilon}_{k+1|k})\,\vec{o} \\ 0 & \end{bmatrix}\right)\right)^{\vee}_{\mathrm{G}'\ |\epsilon=0},
\end{aligned}
\tag{4.14}
$$

where $\mathcal{K}^0_{s_i}(X^{\epsilon}_{k+1|k}) = \mathcal{K}^0_{s_i}\left(X_{k+1|k}\exp_{\mathrm{G}}\left(\epsilon^{\wedge}_{\mathrm{G}}\right)\right)$ corresponds to the forward kinematics for the infinitesimally perturbed state $X_{k+1|k}$. Note that the term $\mathcal{K}^0_{s_i}(X_{k+1|k})^{-1}$ vanishes after applying the partial derivatives over ϵ.

We now decompose the kinematics term $\mathcal{K}^0_s$ into three parts as

$$
\mathcal{K}^0_s = \mathcal{K}^0_l \theta^l_{k+1|k} \mathcal{K}^{l+1}_s\,, \tag{4.15}
$$

where $\mathcal{K}^0_l$ represents the transformation from the base frame to joint l, $\theta^l_{k+1|k}$ denotes current state of joint l and $\mathcal{K}^{l+1}_s$ represents the transformation from joint $l+1$ to marker s. Then, by exploiting results from [103], we can complete the Lie group Jacobian of marker measurement with respect to joint state.

$$
{}^{m}\mathcal{H}^{\theta,l,r}_{k+1} = \mathcal{K}^0_l \theta^l_{k+1|k} E^{l,r} \mathcal{K}^{l+1}_s\,\vec{o}\,, \tag{4.16}
$$

where $E^{l,r}$ represents the r-th generator of a Lie group representing the l-th joint. The last row of the right-hand side term of (4.16) consists of the unit element in homogeneous coordinates and hence only the first three elements are put into $\mathcal{H}^{\theta,l,r}_{k+1}$, while the unit element is removed. Since marker position measurements are only a function of the joint positions, the part of the $\mathcal{H}_{k+1}$ matrix relating measurements with velocity and acceleration components is filled with zero values, i.e., ${}^{m}\mathcal{H}^{\omega,l}_{k+1} = 0$ and ${}^{m}\mathcal{H}^{\alpha,l}_{k+1} = 0$.

4.1.5 Gyro update

The measurement function of the gyro measurement is:

$$
h_g(X_{k+1|k}) = \sum_{i=1}^{n} R^s_i\,\vec{\omega}^i_{k+1|k}\,, \tag{4.17}
$$

where n is the number of joints preceding the gyro sensor s. The term $R_i^s = R_i^s(X_{k+1|k})$ represents the rotational component of the forward kinematics between the i-th joint and the gyro sensor s, thus affecting its measurement [139]. The gyro measurements are affected by position (through kinematics) and velocity, hence the corresponding parts of $\mathcal{H}_{k+1}$ matrix need to be evaluated.

By applying partial derivatives and evaluating the multivariate limits similarly to the marker measurement, the part of $\mathcal{H}_{k+1}$ relating the gyro measurement to the orientation of the l-th joint ${}^g\mathcal{H}_{k+1}^{\theta,l}$ is given as:

$$ {}^g\mathcal{H}_{k+1}^{\theta,l,r} = \sum_{i=1}^{l-1} R_l^s E^{l,r^{\mathrm{T}}} \theta_{k+1|k}^{l^{-1}} R_i^l \vec{\omega}_{k+1|k}^i \,, \tag{4.18} $$

where R_i^l represents the rotation between the i-th and l-th joint, $\theta_{k+1|k}^l$ is the position of the l-th joint. Each of the generators E represents an infinitesimal motion in one of the directions of a Lie group.

The part of $\mathcal{H}_{k+1}$ relating the gyro measurement to the velocity of the l-th joint ${}^g\mathcal{H}_{k+1}^{\omega,l}$ is given as:

$$ {}^g\mathcal{H}_{k+1}^{\omega,l} = R_l^s \,. \tag{4.19} $$

Since gyro measurement (4.17) is not a function of the joint accelerations, the part of the $\mathcal{H}_{k+1}$ matrix relating gyro measurements to l-th joint acceleration components is filled with zero values; ${}^g\mathcal{H}_{k+1}^{\alpha,l} = 0$.

4.1.6 Accelerometer update

The measurement function corresponding to the accelerometer measurement is:

$$ h_a(X_{k+1|k}) = \overbrace{R_0^s \ddot{p}_{k+1|k}}^{\text{point acceleration}} + \overbrace{R_0^s g}^{\text{gravity component}} \,, \tag{4.20} $$

where the first term emerges due to the body motion, while the second term arises due to gravity. The term R denotes that only the rotation part is embedded into an SE(3) member, while the translation part is set to 0. The term $\ddot{p}_{k+1|k}$ represents an acceleration of the sensor s represented in the base frame and given in homogeneous coordinates, while g is the gravity vector in homogeneous coordinates. In order to evaluate $\ddot{p}_{k+1|k}$, we start from defining the IMU position as

$$ p_{k+1|k} = \mathcal{K}_s^0 \vec{o} \,. \tag{4.21} $$

The forward kinematics can be decomposed as

$$\mathcal{K}_s^0 = T_1^0 \theta_{k+1|k}^1 T_2^1 \theta_{k+1|k}^2 \cdots T_n^{n-1} \theta_{k+1|k}^n. \tag{4.22}$$

Each part of the forward kinematics $\mathcal{K}_i^{i-1} = T_i^{i-1}\theta_{k+1|k}^i$ consists of the constant transformation T_i^{i-1} and the position of the i-th joint $\theta_{k+1|k}^i$. We now evaluate the first two derivatives of sensor position $p_{k+1|k}$. The velocity of the point $p_{k+1|k}$ evaluates to

$$\dot{p}_{k+1|k} = \sum_{i=1}^{n} \left(\mathcal{K}_i^0 S_{k+1|k}^{i,\omega} \mathcal{K}_s^i \right) \vec{o}\,, \tag{4.23}$$

where the summation iterates over n joints affecting sensor s, while the term $S_{k+1|k}^{i,\omega}$ is given as

$$S_{k+1|k}^{i,\omega} = \sum_{r=1}^{d_i} \left(\omega_{k+1|k}^{i,r} E^{i,r} \right), \tag{4.24}$$

which is a function of the number of degrees of freedom d_i of the i-th joint, while the superscript ω denotes that the velocity components are summed up. The acceleration of the point $p_{k+1|k}$ evaluates to

$$\begin{aligned} \ddot{p}_{k+1|k} = & \overbrace{\sum_{i=1}^{n} \left(\sum_{j=1}^{i} \left(\mathcal{K}_j^0 S_{k+1|k}^{j,\omega} \mathcal{K}_i^j \right) S_{k+1|k}^{i,\omega} \mathcal{K}_s^i \right) \vec{o}}^{\text{centripetal force component I}} + \\ & \overbrace{\sum_{i=1}^{n} \left(\mathcal{K}_i^0 S_{k+1|k}^{i,\omega} \sum_{j=i+1}^{n} \left(\mathcal{K}_j^i S_{k+1|k}^{j,\omega} \mathcal{K}_s^j \right) \right) \vec{o}}^{\text{centripetal force component II}} + \\ & \underbrace{\sum_{i=1}^{n} \left(\mathcal{K}_i^0 S_{k+1|k}^{i,\alpha} \mathcal{K}_s^i \right) \vec{o}}_{\text{joint acceleration component}} \quad . \end{aligned} \tag{4.25}$$

The acceleration $\ddot{p}_{k+1|k}$ consists of two components – the centripetal force component and the joint acceleration component, which we emphasize in (4.25).

We now proceed to linearize and evaluate the part of $\mathcal{H}_{k+1}$ corresponding to the accelerometer measurement and joint l:

$${}^a\mathcal{H}_{k+1}^l = \frac{\partial R_0^s}{\partial X_{k+1|k}^l} \left(\ddot{p}_{k+1|k} + g \right) + R_0^s \frac{\partial \ddot{p}_{k+1|k}}{\partial X_{k+1|k}^l}\,. \tag{4.26}$$

In order to evaluate (4.26) we need to compute partial derivatives of R_0^s and $\ddot{p}_{k+1|k}$ with respect to position ${}^a\mathcal{H}_{k+1}^{\theta,l}$, velocity ${}^a\mathcal{H}_{k+1}^{\omega,l}$, and acceleration ${}^a\mathcal{H}_{k+1}^{\alpha,l}$.

4.1.6.1 Positional part ${}^a\mathcal{H}_{k+1}^{\theta,l}$

We start by evaluating the partial derivative of forward kinematics R_0^s with respect to the positional variable $\theta_{k+1|k}^{l,r}$, where r relates to the r-th generator, $r = 1, .., d_l$, with d_l being the number of degrees of freedom of joint l. This evaluates to

$$\frac{\partial R_0^s}{\partial \theta_{k+1|k}^{l,r}} = R_0^l \theta_{k+1|k}^{l,r} E^r R_l^s \,, \tag{4.27}$$

where $E^{l,r}$ represents the r-th generator of a Lie group representing the l-th joint. The evaluation of the partial derivative of acceleration $\ddot{p}_{k+1|k}$ with respect to the positional variable $\theta_{k+1|k}^{l,r}$ evaluates to

$$\begin{aligned}
\frac{\partial \ddot{p}_{k+1|k}}{\partial \theta_{k+1|k}^{l,r}} = & \sum_{i=1}^{n} \left(\sum_{j=1}^{i} \begin{Bmatrix} \mathcal{K}_l^0 \, E^{l,r} \, \mathcal{K}_j^l \, S_{k+1|k}^{j,\omega} \, \mathcal{K}_i^j \, S_{k+1|k}^{i,\omega} \, \mathcal{K}_s^i \,, & l \leq j \\ \mathcal{K}_j^0 \, S_{k+1|k}^{j,\omega} \, \mathcal{K}_l^j \, E^{l,r} \, \mathcal{K}_i^l \, S_{k+1|k}^{i,\omega} \, \mathcal{K}_s^i \,, & j < l \leq i \\ \mathcal{K}_j^0 \, S_{k+1|k}^{j,\omega} \, \mathcal{K}_i^j \, S_{k+1|k}^{i,\omega} \, \mathcal{K}_l^i \, E^{l,r} \, \mathcal{K}_s^l \,, & i < l \end{Bmatrix} \right) \vec{o} + \\
& \sum_{i=1}^{n} \left(\sum_{j=i+1}^{n} \begin{Bmatrix} \mathcal{K}_l^0 \, E^{l,r} \, \mathcal{K}_i^l \, S_{k+1|k}^{i,\omega} \, \mathcal{K}_j^i \, S_{k+1|k}^{j,\omega} \, \mathcal{K}_s^j \,, & l \leq i \\ \mathcal{K}_i^0 \, S_{k+1|k}^{i,\omega} \, \mathcal{K}_l^i \, E^{l,r} \, \mathcal{K}_j^l \, S_{k+1|k}^{j,\omega} \, \mathcal{K}_s^j \,, & i < l \leq j \\ \mathcal{K}_i^0 \, S_{k+1|k}^{i,\omega} \, \mathcal{K}_j^i \, S_{k+1|k}^{j,\omega} \, \mathcal{K}_l^j \, E^{l,r} \, \mathcal{K}_s^l \,, & j < l \end{Bmatrix} \right) \vec{o} + \\
& \sum_{i=1}^{n} \begin{Bmatrix} \mathcal{K}_l^0 \, E^{l,r} \, \mathcal{K}_i^l \, S_{k+1|k}^{i,\alpha} \, \mathcal{K}_s^i \,, & l \leq i \\ \mathcal{K}_i^0 \, S_{k+1|k}^{i,\alpha} \, \mathcal{K}_l^i \, E^{l,r} \, \mathcal{K}_s^l \,, & i < l \end{Bmatrix} \vec{o} \,,
\end{aligned} \tag{4.28}$$

where

$$S_{k+1|k}^{i,\omega} = \sum_{r=1}^{d_i} \left(\omega_{k+1|k}^{i,r} E^{i,r} \right), \tag{4.29}$$

which is a function of the number of degrees of freedom d_i of the i-th joint, and the superscript ω denotes that the velocity components are summed up. The three parts in (4.28) arise from evaluating partial derivatives of the three components existing in equation 4.25, i.e., the two centripetal components and the joint acceleration component. Depending on the location within kinematic chain of the considered joint l, different terms need to be applied. This completes the derivation positional component of equation 4.26.

4.1.6.2 Velocity part ${}^{a}\mathcal{H}^{\omega,l}_{k+1}$

Since R^s_0 is only a function of the joint position $\theta^l_{k+1|k}$, the partial derivative of forward kinematics with respect to the velocity component is

$$\frac{\partial R^s_0}{\partial \omega^{l,r}_{k+1|k}} = 0\,. \tag{4.30}$$

We now evaluate the partial derivative of acceleration $\ddot{p}_{k+1|k}$, with respect to the velocity variable $\omega^{l,r}_{k+1|k}$, which evaluates to the following expression

$$\begin{aligned}\frac{\partial \ddot{p}_{k+1|k}}{\partial \omega^{l,r}_{k+1|k}} = \mathcal{K}^0_l E^{l,r} \sum_{i=l}^{n} \left(\mathcal{K}^l_i S^{i,\omega}_{k+1|k} \mathcal{K}^i_s\right)\vec{o} \;+\; \sum_{j=1}^{l} \left(\mathcal{K}^0_j S^{j,\omega}_{k+1|k} \mathcal{K}^j_l\right) E^{l,r} \mathcal{K}^l_s \vec{o} \;+ \\ \mathcal{K}^0_l E^{l,r} \sum_{j=l+1}^{n} \left(\mathcal{K}^l_j S^{j,\omega}_{k+1|k} \mathcal{K}^j_s\right)\vec{o} \;+\; \sum_{i=1}^{l-1} \left(\mathcal{K}^0_i S^{i,\omega}_{k+1|k} \mathcal{K}^i_l\right) E^{l,r} \mathcal{K}^l_s \vec{o}\,.\end{aligned} \tag{4.31}$$

The four parts of this derivative arise from the two centripetal force components (two per each) given in equation 4.26. The complete velocity component can now be calculated as

$$\begin{bmatrix} {}^{a}\mathcal{H}^{\omega,l,r}_{k+1} \\ 1 \end{bmatrix} = \frac{\partial \mathcal{K}^{s,R}_0}{\partial \omega^{l,r}_{k+1|k}} \left(\ddot{p}_{k+1|k} + g\right) + \mathcal{K}^{s,R}_0 \frac{\partial \ddot{p}_{k+1|k}}{\partial \omega^{l,r}_{k+1|k}}\,. \tag{4.32}$$

4.1.6.3 Acceleration part ${}^{a}\mathcal{H}^{\alpha,l}_{k+1}$

Here, we evaluate the acceleration term. Similar to the velocity term, derivative of forward kinematics with respect to the acceleration component is

$$\frac{\partial R^s_0}{\partial \alpha^{l,r}_{k+1|k}} = 0\,. \tag{4.33}$$

The partial derivative of acceleration $\ddot{p}_{k+1|k}$ with respect to the r-th component of the acceleration of the l-th joint, $\alpha^{l,r}_{k+1|k}$, evaluates as

$$\frac{\partial \ddot{p}_{k+1|k}}{\partial \alpha^{l,r}_{k+1|k}} = \mathcal{K}^0_l E^{l,r} \mathcal{K}^l_s \vec{o}\,. \tag{4.34}$$

This derivative arise from the joint acceleration component given in equation (27) of the original manuscript. The complete acceleration component can now be calculated as

$$\begin{bmatrix} {}^{a}\mathcal{H}^{\alpha,l,r}_{k+1} \\ 1 \end{bmatrix} = \frac{\partial R^{s}_{0}}{\partial \alpha^{l,r}_{k+1|k}} \left(\ddot{p}_{k+1|k} + g \right) + R^{s}_{0} \frac{\partial \ddot{p}_{k+1|k}}{\partial \alpha^{l,r}_{k+1|k}} \,. \tag{4.35}$$

Finally, the full ${}^{a}\mathcal{H}^{l}_{k+1}$ relating sensor measurement and the system variables is constructed.

$${}^{a}\mathcal{H}^{l}_{k+1} = \begin{bmatrix} \mathcal{H}^{\theta,l}_{k+1} & \mathcal{H}^{\omega,l}_{k+1} & \mathcal{H}^{\alpha,l}_{k+1} \end{bmatrix} \,. \tag{4.36}$$

4.2 Observability analysis

A system is observable if its state at a certain time instant can be uniquely determined given a finite sequence of its input and outputs [140]. In the case of an estimation problem dealing with the kinematic model of the human body, system observability translates to the property that the joint states can be determined based on the measurements, e.g., markers or IMUs. Practically, observability provides information about the minimal sufficient measurement setup needed for determining all the joint states of interest. In the following, we focus on the joint orientation observability, since joint velocities and accelerations are observable via direct dependency, once it is shown that the orientation is observable.

Approaches for observability analysis of linear time-invariant systems include well-established tests including the rank of the Gramian matrix [141] or the Popov–Belevitch–Hautus (PBH) test [142]. The PBH test cannot standardly be applied to linearized systems because of the time-invariance requirement, although some recent generalizations of the PBH explore the use of nonlinear eigenvalues [143]. The test based on the Gramian matrix could theoretically be applied, but practically becomes intractable for 6dof applications; hence, evaluation via the Gramian matrix is usually performed numerically [144].

The observability analysis in this work relies on a differential geometric characterization of observability, which leads to the evaluation of the observability rank condition based on Lie derivatives [145]. This approach has been used in various applications, beginning with the observability of a map-based single robot localization [146] and cooperative localization of pairs of robots [147] in 2D, and subsequently with a full SLAM system [148] and camera-odometry extrinsic calibration in 2D [149]. Although early applications dealt with 2D state space, the approach was also successfully applied for observability properties of camera–IMU extrinsic parameters calibration [144, 150] dealing with 3D applications, i.e., 6dof transformations. Furthermore, in [151] the same approach is used for evaluating estimator inconsistency in a vision aided inertial navigation system.

4.2.1 Nonlinear observability

In [145], the state space is assumed to be a smooth manifold and the considered non-linear system has the following form

$$\begin{cases} \dot{x} = f(x,u) \\ z = h(x) \end{cases}, \tag{4.37}$$

where $\boldsymbol{x} \in \mathbb{R}^p$ is the state vector, $z \in \mathbb{R}^q$ is the measurement vector, u is the control input, $f(\cdot)$ is the nonlinear system state equation, and $h(\cdot)$ is the nonlinear measurement equation. Some previous works relied on incorporating a subset of measurements through the input signal [144, 150, 151], while in our approach we assume zero control inputs, i.e., $u = 0$. The zeroth-, first- and second-order Lie derivatives L of the function h with respect to f at x are formed recursively as follows

$$\begin{aligned} L^0 h_s(x) &= h_s(x) \\ L_f^1 h_s(x) &= \nabla h_s(x) \cdot f(x) \\ L_f^2 h_s(x) &= L_f^1\Big(L_f^1 h_s(x)\Big) = \nabla L_f^1 h_s(x) \cdot f(x) \\ &\vdots \\ L_f^{i+1} h_s(x) &= \nabla L_f^i h_s(x) \cdot f(x), \end{aligned} \tag{4.38}$$

where '$\cdot$' denotes the vector inner product, s represents the measurement of the s-th sensor, and '∇' is the gradient operator (partial derivative). Then, the observability matrix is defined as the matrix with rows

$$\mathcal{O} = \left\{ \nabla L_f^l h_s(x) \,\middle|\, s = 1, ..., m; l \in \mathbb{N} \right\}, \tag{4.39}$$

where l is the order of the Lie derivative. If $\mathcal{O}$ is full rank, the system is locally weakly observable [144].

4.2.2 Nonlinear observability on Lie groups

Since Lie algebra is tangential to the pertaining Lie group at every point of the state space, analysis leading to locally weak observability can be considered in the Lie algebra space without loss of generality. Therefore, similarly to the derivations of the Jacobians in Secs. 4.1.2 and

4.1.3, instead of evaluating gradients directly over the variables on the group, we rather re-define the problem such that the system is described in terms of local Lie algebra coordinates of the current state X. The continuous-time system used for the analysis is then given as

$$\begin{cases} \dot{x} = \Omega(X) \\ z = \log^{\vee}_{\mathrm{G}'}\left(h(X)\right) \end{cases}, \tag{4.40}$$

where the motion model $\Omega(X)$ is given in the Lie algebra local coordinates. The zeroth-, first- and second-order Lie derivatives $\mathcal{L}$ used for the Lie group based formulation of the system, with measurement function h with respect to Ω at X are then given as

$$\begin{aligned} \mathcal{L}^0\, h_s(X) &= \log^{\vee}_{\mathrm{G}'}\left(h_s(X)\right) \\ \mathcal{L}^1_{\Omega}\, h_s(X) &= \nabla_{\mathrm{G}} \log^{\vee}_{\mathrm{G}'}\left(h_s(X)\right) \cdot \Omega(X) \\ &= \frac{\partial \log^{\vee}_{\mathrm{G}'}\left(h_s\left(X \exp^{\wedge}_{\mathrm{G}}(\xi)\right)\right)}{\partial \xi}\bigg|_{\xi=0} \cdot \Omega(X) \\ \mathcal{L}^2_{\Omega}\, h_s(X) &= \mathcal{L}^1_{\Omega}\Big(\mathcal{L}^1_{\Omega}\, h_s(X)\Big) = \nabla_{\mathrm{G}} \mathcal{L}^1_{\Omega}\, h_s(X) \cdot \Omega(X) \\ &\vdots \\ \mathcal{L}^{i+1}_{\Omega} h_s(x) &= \nabla_{\mathrm{G}} \mathcal{L}^i_{\Omega}\, h_s(x) \cdot \Omega(X), \end{aligned} \tag{4.41}$$

where ∇_{G} is a gradient operator such that some noisy perturbation ξ is added to the current state and then partial derivatives are evaluated as ξ approaches 0. Note that a similar idea was used for the evaluation of Lie group Jacobians (4.7) and (4.11). Based on these expressions, the observability matrix is defined with rows

$$\mathcal{O} = \left\{ \nabla_{\mathrm{G}} \mathcal{L}^l_{\Omega}\, h_s(X) \,\middle|\, s = 1, ..., m;\ l \in \mathbb{N} \right\}. \tag{4.42}$$

Again, if $\mathcal{O}$ is full rank, the system is locally weakly observable. Both matrices (4.47) and (4.42) can have an infinite number of rows; however, to prove that they are full rank, it is sufficient to show that a subset of rows are linearly independent [144]. There exists no systematic method for selecting suitable Lie derivatives for designing the observability matrix. Hence, this matrix is formed by sequentially considering directions of the state space along which the gradient of each of the candidate Lie derivatives provides information [144].

4.2.3 Marker based observability

We now consider observability of joint states relying on markers attached to an arbitrarily long chain of SO(3) joints. An illustration of a general SO(3) joints chain is given in Fig. 4.2, where all

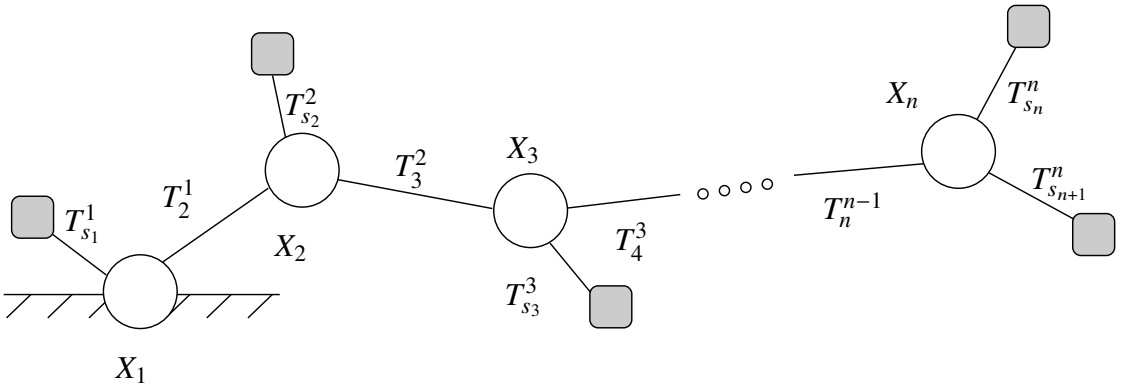


Figure 4.2 Illustration of an SO(3) joints chain. A single marker is attached to each joint, while two markers are attached to the last joint.

but the last joint have a single marker attached, while the last joint has two markers. In the proof of the observability of this system, we will need only the zeroth-order Lie derivative $\mathcal{L}^0 h_{s_i}(X)$

$$\nabla^{\theta}_{\mathrm{G}} \mathcal{L}^0 h_{s_i}(X) = \frac{\partial \mathcal{L}^0 h_{s_i}\left(X \exp^{\wedge}_{\mathrm{G}}(\xi)\right)}{\partial \xi}\Big|_{\xi=0} = \mathcal{H}_{s_i}\,, \tag{4.43}$$

where $\mathcal{H}_{s_i}$ is a matrix Jacobian evaluated using (4.16), and $X \in \mathrm{SE}(3) \times \cdots \times \mathrm{SE}(3)$ with all translational components set to zero. If the considered marker s_i is affected by the joint j, the respective Jacobian is given as

$$\mathcal{H}^j_{s_i} = \mathcal{K}^0_j X_j E(\mathcal{K}^j_{s_i} \vec{o}) \tag{4.44}$$

and otherwise $\mathcal{H}^j_{s_i} = 0$. The operator $E(\cdot)$ is constructed so that $\mathfrak{so}(3)$ generators are embedded into $\mathfrak{se}(3)$ in the positions of rotational generators, while the translational generators of $\mathfrak{se}(3)$ are set to zero:

$$E(\vec{t}) = \begin{bmatrix} & & & 0 \\ E_x t & E_y t & E_z t & 0 \\ & & & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \in \mathfrak{se}(3)\,, \tag{4.45}$$

where $\vec{t}$ are homogeneous coordinates, t are its first three elements, and E_x, E_y, E_z are rotational generators of $\mathfrak{so}(3)$.

We re-write (4.44) by applying properties of the operator $E(\cdot)$ as follows

$$\begin{aligned} \mathcal{H}^j_{s_i} &= \mathcal{K}^0_j X_j E(\mathcal{K}^j_{s_i} \vec{o}) = R^0_j X_j E(\mathcal{K}^j_{s_i} \vec{o}) \\ &= E(R^0_j X_j \mathcal{K}^j_{s_i} \vec{o}) R^0_j X_j\,. \end{aligned} \tag{4.46}$$

We first applied the property that $\mathcal{K}E(\vec{t}) = RE(\vec{t})$, where $\mathcal{K} \in \mathrm{SE}(3)$ is the full kinematics and $R \in \mathrm{SE}(3)$ is the rotational kinematics with the translation elements set to zero. This property holds due to the form of $E(\cdot)$, i.e., the translation components of all the transformation matrices preceding E can be discarded and filled with zeros, since the last row/column of $E(\cdot)$ are zero, hence eliminating translational components via multiplication. Second, we apply the property $RE(\vec{t}) = E(R\vec{t})R$, since because of the construction of X the product $R_j^0 X_j$ represents rotation with zero translational components.

The observability matrix $\mathcal{O}$ is constructed such that $n+1$ is the number of markers, corresponding to $n+1$ rows, and n is the number of joints, corresponding to n columns:

$$\mathcal{O} = \begin{bmatrix} \mathcal{H}^1_{s_1} & 0 & \cdots & 0 \\ \mathcal{H}^1_{s_2} & \mathcal{H}^2_{s_2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \mathcal{H}^1_{s_n} & \mathcal{H}^2_{s_n} & \cdots & \mathcal{H}^n_{s_n} \\ \mathcal{H}^1_{s_{n+1}} & \mathcal{H}^2_{s_{n+1}} & \cdots & \mathcal{H}^n_{s_{n+1}} \end{bmatrix}. \tag{4.47}$$

The observability matrix contains non-zero elements whenever a joint motion affects sensor measurements. Otherwise, a zero element appears if a joint motion does not affect the considered marker measurement. For example, by considering Fig. 4.2, we can see that motion of joint X_1 affects the measurements of all the markers, while motion of the joint X_3 affects only measurements of the last two markers.

In order to prove that (4.47) is full rank, we now proceed with Gaussian elimination. By inserting the last expression of (4.46), the observability matrix becomes

$$\mathcal{O} = \begin{bmatrix} E(R_1^0 X_1 \mathcal{K}^1_{s_1} \vec{o}) R_1^0 X_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ E(R_1^0 X_1 \mathcal{K}^1_{s_n} \vec{o}) R_1^0 X_1 & \cdots & E(R_n^0 X_n \mathcal{K}^n_{s_n} \vec{o}) R_n^0 X_n \\ E(R_1^0 X_1 \mathcal{K}^1_{s_{n+1}} \vec{o}) R_1^0 X_1 & \cdots & E(R_n^0 X_n \mathcal{K}^n_{s_{n+1}} \vec{o}) R_n^0 X_n \end{bmatrix}$$

Since right multiplication of the whole column by the same transformation only performs a linear combination of row elements, we multiply the i-th column with $\left(R_i^0 X_i\right)^{-1}$ elements outside the brackets obtaining

$$\mathcal{O} = \begin{bmatrix} E(R_1^0 X_1 \mathcal{K}^1_{s_1} \vec{o}) & \cdots & 0 \\ \vdots & \ddots & \vdots \\ E(R_1^0 X_1 \mathcal{K}^1_{s_n} \vec{o}) & \cdots & E(R_n^0 X_n \mathcal{K}^n_{s_n} \vec{o}) \\ E(R_1^0 X_1 \mathcal{K}^1_{s_{n+1}} \vec{o}) & \cdots & E(R_n^0 X_n \mathcal{K}^n_{s_{n+1}} \vec{o}) \end{bmatrix}. \tag{4.48}$$

In order to proceed further with simplifying the observation matrix for Gaussian elimination, we look next at how we can decompose the kinematic transformations. Kinematics between the i-th joint, preceding the j-th sensor, is given as

$$\mathcal{K}^i_{s_j} = T^i_{i+1} X_{i+1} \cdots T^{j-1}_j X_j T^j_{s_j} \,. \tag{4.49}$$

It follows that

$$\begin{aligned} E\big(R^0_i X_i \mathcal{K}^i_{s_j} \vec{o}\,\big) &= E\big(R^0_i X_i T^i_{i+1} X_{i+1} \cdots X_{j-1} T^{j-1}_j X_j T^j_{s_j} \vec{o}\big) \\ &= E\big(R^0_i X_i \vec{t}^{\,i}_{i+1} + \cdots + R^0_{j-1} X_{j-1} \vec{t}^{\,j-1}_j + R^0_j X_j \vec{t}^{\,j}_{s_j}\big) \\ &= E\Big(\sum_{k=i}^{j-1} R^0_k X_k \vec{t}^{\,k}_{k+1} + R^0_j X_j \vec{t}^{\,j}_{s_j}\Big) \\ &= E\Big(\sum_{k=i}^{j-1} \tau^k_{k+1} + \tau^j_{s_j}\Big), \end{aligned} \tag{4.50}$$

where $\vec{t}^{\,i}_{i+1}$ represents the translation component of transformation matrix T^i_{i+1}, and $\tau^k_{k+1} = R^0_k X_k \vec{t}^{\,k}_{k+1}$. We continue the Gaussian elimination procedure by writing the observability matrix elements in terms of sums as in (4.50), and consider the upper left corner of the matrix

$$\begin{bmatrix} E(\tau^1_{s_1}) & 0 & 0 & \cdots \\ E(\tau^1_2 + \tau^2_{s_2}) & E(\tau^2_{s_2}) & 0 & \cdots \\ E(\tau^1_2 + \tau^2_3 + \tau^3_{s_3}) & E(\tau^2_3 + \tau^3_{s_3}) & E(\tau^3_{s_3}) & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}. \tag{4.51}$$

Since it holds that $E(a + b) = E(a) + E(b)$, by applying several simple column and row subtractions, we obtain

$$\begin{bmatrix} E(\tau^1_{s_1}) & 0 & 0 & 0 & \cdots \\ E(\tau^1_2) & E(\tau^2_{s_2}) & 0 & 0 & \cdots \\ 0 & E(\tau^2_3 - \tau^2_{s_2}) & E(\tau^3_{s_3}) & 0 & \cdots \\ 0 & 0 & E(\tau^3_4 - \tau^3_{s_3}) & E(\tau^4_{s_4}) & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix} = A \,.$$

We now proceed by evaluating the nullspace of matrix A, i.e., to determine if there exists an $x \neq 0$ such that

$$Ax = 0 \,. \tag{4.52}$$

which would indicate that A is not full rank. For this purpose we extract the 2-nd and 3-rd rows of the A matrix, and separate vector x into components as follows

$$\begin{bmatrix} E(\tau_2^1) & E(\tau_{s_2}^2) & 0 & 0 & \cdots \\ 0 & E(\tau_3^2 - \tau_{s_2}^2) & E(\tau_{s_3}^3) & 0 & \cdots \end{bmatrix} \begin{bmatrix} x_{1:3} \\ x_{4:6} \\ x_{7:9} \\ \vdots \end{bmatrix} = 0\,. \tag{4.53}$$

The analysis can then be generalized to other pairs of rows given the symmetry of A. The part of the x vector that corresponds to the 4-th to 6-th elements is

$$\begin{bmatrix} E(\tau_{s_2}^2) \\ E(\tau_3^2 - \tau_{s_2}^2) \end{bmatrix} x_{4:6} = 0\,. \tag{4.54}$$

We can observe that there is no solution $x_{4:6} \neq 0$ unless $\tau_{s_2}^2 = \alpha\tau_3^2$, where $\alpha \neq 0$. A similar observation then applies to the entire A and x, meaning that A is full rank except for some special cases that are further discussed below.

4.2.4 Three joint example

Using the results given in (4.54), we consider observability of a three joint chain emulating a human arm or leg. The illustration of the three joint chain is given in Fig. 4.2, when $n = 3$. The complete matrix A for this system evaluates to

$$\begin{bmatrix} E(\tau_{s_1}^1) & 0 & 0 \\ E(\tau_2^1) & E(\tau_{s_2}^2) & 0 \\ 0 & E(\tau_3^2 - \tau_{s_2}^2) & E(\tau_{s_3}^3) \\ 0 & E(\tau_3^2 - \tau_{s_2}^2) & E(\tau_{s_4}^3) \end{bmatrix}. \tag{4.55}$$

By decomposing terms, as in the previous section, we identify the conditions that lead to observability violations

- the terms $\tau_{s_1}^1$ and τ_2^1 correspond to

  $$\tau_{s_1}^1 = R_1^0 X_1 t_{s_1}^1 \text{ and } \tau_2^1 = R_1^0 X_1 t_2^1\,,$$

  hence observability would be violated if $t_{s_1}^1 = \alpha t_2^1$, i.e., if $t_{s_1}^1$ and t_2^1 are colinear, which occurs if sensor s_1 is placed on the axis defined by connection of joints 1–2.

Analogous results can be obtained relating sensor s_2 and the connection between joints 2 and 3, and relating the axes of sensors s_3 and s_4. Generally, we can conclude that the observability is violated if a marker is placed on the axis connecting its associated joint with the succeeding one.

4.3 Simulation Validation

The proposed approach is extensively validated in simulation. We first compare the proposed approach to estimation utilizing a standard kinematic model defined with revolute and prismatic joints [152], as shown in Fig. 4.1 (left). We discuss the gimbal lock limitation of modeling kinematics with Euler angles and show that because Lie group based models do not have gimbal lock, LG-EKF not only provides better estimation but is also easier to tune for best performance. To ensure that the estimation improvement is due to the difference in representation and not selected filtering method we employ the Unscented Kalman filter (UKF) in addition to EKF when estimating gimbal lock Euler angles. UKF propagates carefully chosen samples through the non linear state update and measurement functions to estimate the mean, covariance, and cross correlation of the state and observation. It captures the mean and covariance up to second order and does not require explicit calculation of the Jacobians [153]. The state of the EKF and UKF is defined as the position q, velocity $\dot{q}$, and acceleration $\ddot{q}$ of the joints. Just as in LG-EKF, the constant acceleration model is used.

To compare the estimated rotation of each SO(3) joint with the ground truth, we use the deviation from the identity matrix as the distance metric [154]

$$\mathcal{D}_F = \left\| I - R_e^{\mathrm{T}} R_{gt} \right\|_F , \tag{4.56}$$

where R_e and R_{gt} are the estimated and ground truth rotation matrices of each joint and $\|\cdot\|_F$ denotes the Frobenius norm, which is funcionally equivalent to the geodesic on SO(3) [154]. Subsequently, we verify the observability analysis on a model of a human arm, and demonstrate the sensor setup under which the arm orientation is observable.

4.3.1 Gimbal lock

When the Euler angle kinematic modeling approach is used to represent spherical joints, two of the axes can become aligned and thus a degree of freedom is lost; this is referred to as gimbal lock. In this configuration rotation about the locked axis cannot be estimated. Typically the order of the joint axes is carefully selected to try and avoid the lock, however in human motion estimation, gimbal lock often takes place at the shoulder joint due to its high maneuverability. Unlike the Euler angle formulation, an SO(3) representation of the spherical joint does not suffer from gimbal lock and thus LG-EKF will accurately estimate any rotation.

To demonstrate the benefits of SO(3) representation over Euler angles during gimbal lock we simulate a single spherical joint at the origin with two markers attached at an offset. The

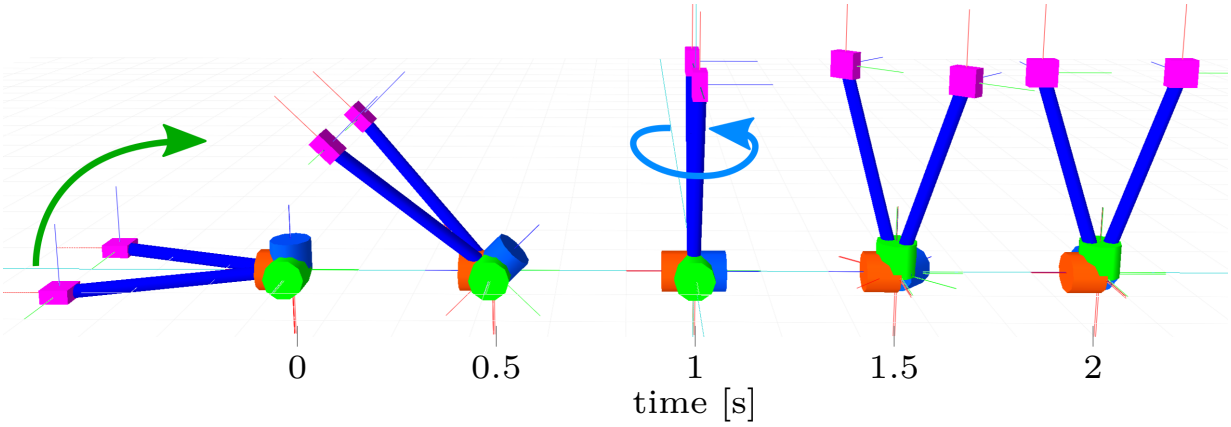


Figure 4.3 Snapshots of the Euler model at different times throughout the gimbal lock validation motion. The green and light blue arrows represent the direction of model rotation about world y and z axis respectively. To ensure the system is observable, two markers (pink boxes) are attached at a fixed offset (illustrated by the blue bar) from a 3dof joint. In Euler angle representation as the second angle (green cylinder) approaches gimbal lock the other two axes (red and blue) align, removing a degree of freedom. To rotate about world z axis the model must first leave gimbal lock, during estimation this typically results in high velocities and increased error in the Euler joints.

simulated Euler angle model and its motion is shown in Fig. 4.3. A quintic polynomial is used to generate a smooth trajectory, sampling at 100 Hz. First, the model experiences a 1 s rotation about the world y axis with initial position 0 rad and final positions $\frac{\pi}{2}$ rad and zero initial and final velocity and acceleration. In the Euler angle model this motion aligns the first and third revolute joint axes putting it into a singularity and removing a degree of freedom (gimbal lock). Next, the model experiences the same 1 s rotation in the now locked world z axis. In order to focus only on the gimbal lock problem, no noise was added to the measurements.

Figure 4.4 shows the error in estimation for the filters. When Euler angles enter gimbal lock, its Jacobian is singular and thus the linearized system is no longer observable. While UKF does not rely on the Jacobian and outperforms EKF, it requires high accelerations in the Euler angles to quickly come out of gimbal lock, during this period the estimation error increases. Furthermore, propagation of the sample points through the non linear measurement function is computationally expensive and it is not feasible to run UKF in real time for full body estimation[2].

Furthermore we show that the Lie group motion model is superior for process noise

[2]For full body (30dof) we would need to compute forward kinematics 181 times at each iteration since the state includes positions, velocities, and accelerations.

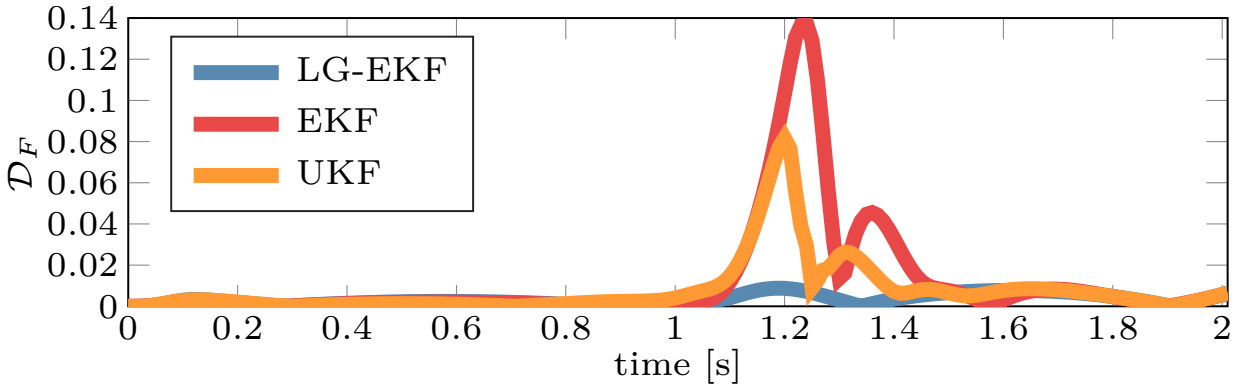


Figure 4.4 LG-EKF, EKF, and UKF estimation during gimbal lock. All filters accurately estimate the rotation about the y axis until the system gets close to the gimbal lock, which happens at 1 second. After the rotation about y the Euler angle model is in gimbal lock and thus EKF and UKF cannot accurately track the orientation until the lock is escaped at 1.2 s. Once Euler angles escape the gimbal lock, EKF and UKF regain an accurate estimate. LG-EKF is unaffected by gimbal lock.

representation over the Euler angle motion model. Consider a single SO(3) joint with a marker or IMU attached at some offset. Independent of the initial SO(3) state, addition of zero mean, Gaussian process noise to the state results in a consistent distribution of the end effector position. With the Euler angle model, adding the same process noise results in end effector position distribution that is state dependent, as illustrated in Fig. 4.5. Thus near gimbal lock Euler EKF requires higher process noise to capture the variability in a highly maneuverable 3D joint such as the shoulder while LG-EKF process noise will remain constant and lower for the entire state space. Thus it should be easier to tune LG-EKF for better performance over the entire state space.

4.3.2 Observability

To verify our observability analysis we simulate the manipulator shown in Fig. 4.2 ($n = 3$) and investigate the convergence properties of LG-EKF in the observable and unobservable cases described in Sec. 4.2. In the zero configuration, the manipulator is standing upright, each link length is 0.5 m. When the system is observable, the markers are attached at offsets $t^1_{s_1}$, $t^2_{s_2}$, $t^3_{s_3}$, $t^3_{s_4}$ of [0.1 0 0.3], [0.1 0 0.25], [0.1 0 0.2], [−0.1 0 0.2] respectively. We allow LG-EKF to converge from the zero configuration to a random static pose, the observation noise, η, and initial covariance are set to 0.01, 1, and identity. Figure 4.6 shows the convergence of the observable case.

Next, the second marker is placed on the axis between joints two and three ($t^2_{s_2} = [0\ 0\ 0.25]$).

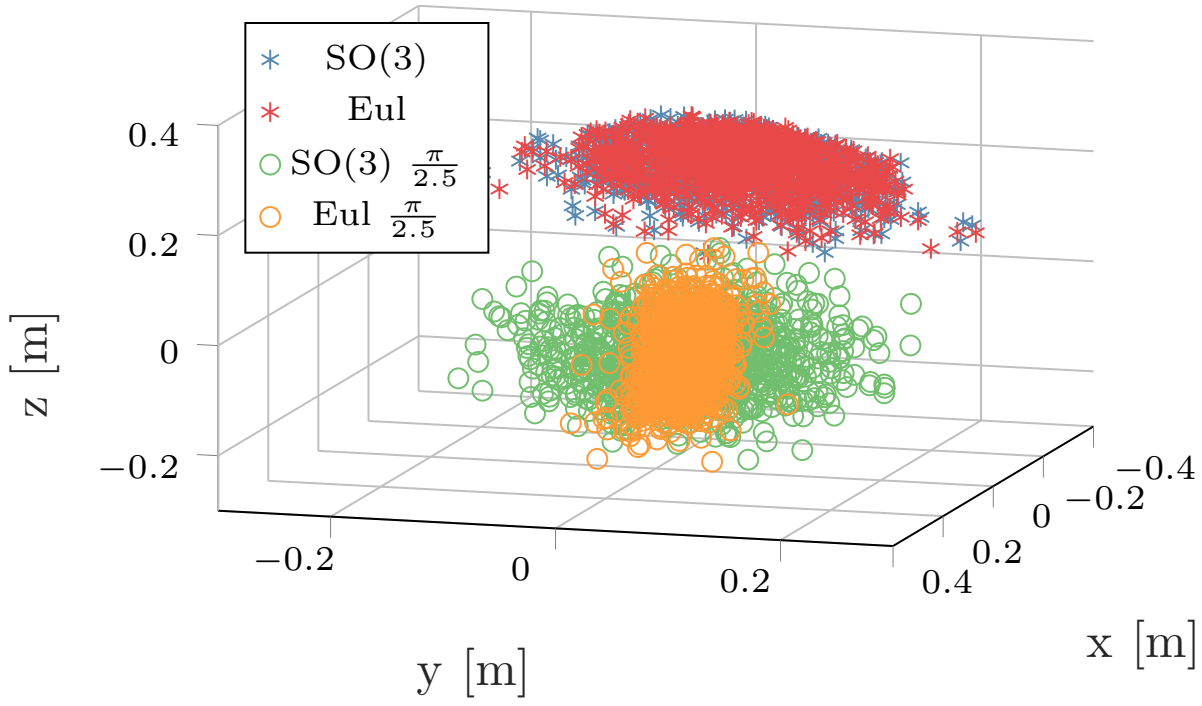


Figure 4.5 An IMU is attached to an SO(3) joint with an offset of [0.1 0.1 0.3] in (x, y, z) axes. Zero mean Gaussian noise with standard deviation of 0.2 is added to the identity state and when the configuration is rotated by $\frac{\pi}{2.5}$ radians. The Lie group representation (blue and green) retains the distribution properties through the rotation about the y axis. In the Euler angle representation the distribution is significantly altered when the axes are no longer perpendicular.

Following the similar discussion as given in subsection 4.2.4, if $t^2_{s_2}$ and t^2_3 are colinear, which occurs if sensor s_2 is placed on the axis defined by connection of joints 2–3, the observability criteria are not satisfied and we do not expect the filter to converge to the true pose. Figure 4.7 shows the convergence of each joint and the LG-EKF measurement residuals in this unobservable case.

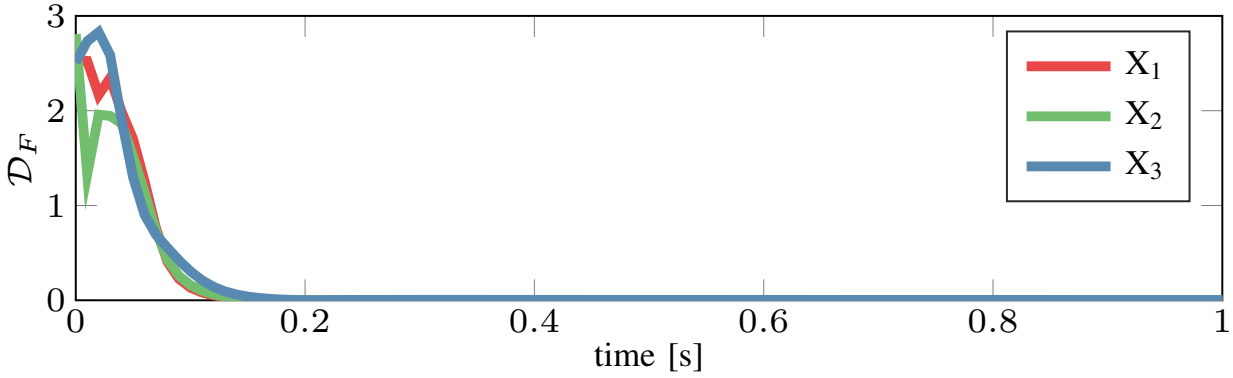


Figure 4.6 LG-EKF correctly converges to the true pose of the manipulator when all of the observability criteria are satisfied

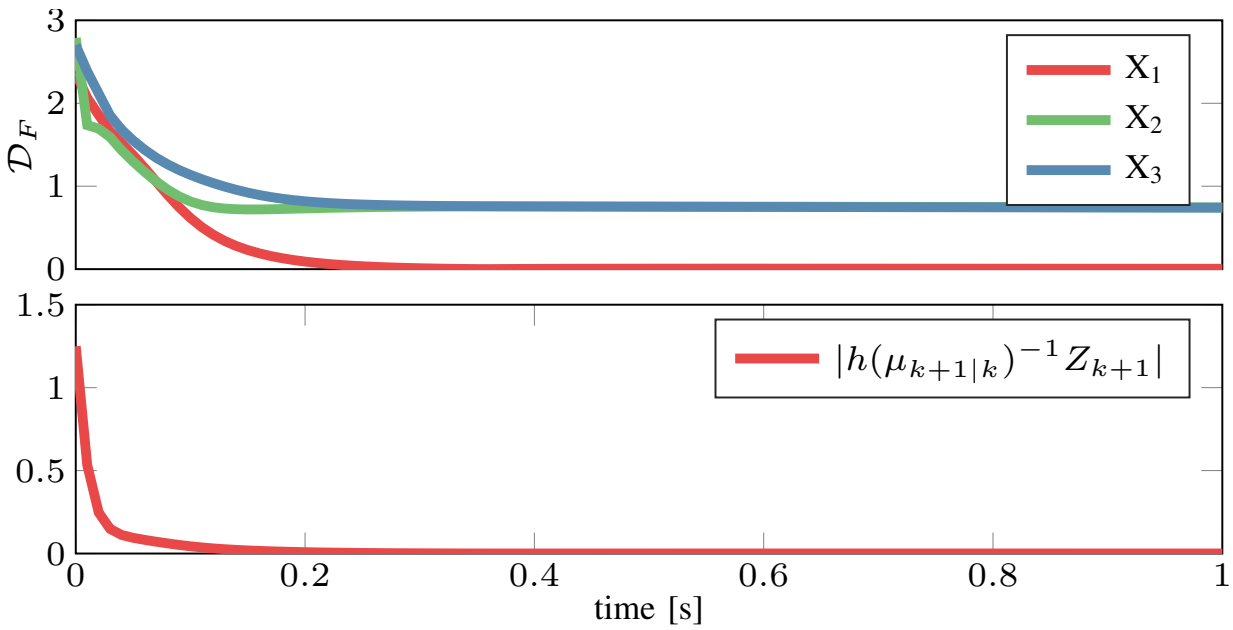


Figure 4.7 When observability criteria is not satisfied, even though the measurement residuals are minimized, LG-EKF does not converge to the true pose. In this particular setup, state estimate of the second joint converges onto an arbitrary rotation about the axis between joints two and three, next to ensure the residuals of markers three and four are minimized this rotation is compensated for by the last joint.

4.4 Real data Validation

First we show that the proposed filtering method is applicable for marker based full body motion capture using real human motion data from the CMU Graphics Lab Motion Capture Database [155]. Next we verify that LG-EKF overcomes gimbal lock which can occur in complex arm movements allowing for accurate pose estimation using only wearable IMUs. Finally, we demonstrate that the consistent distribution of the SO(3) provides a better measure of movement variance.

4.4.1 CMU Dataset

Five different human actions were chosen from the CMU motion capture database [155] to cover a variety of human motions, consisting of boxing (B), dancing (D), running (R), stretching (St), and soccer kick movements (So). Two data sequences, performed by different participants, were chosen for each movement type for a total of 10 data sequences. Movement in the CMU database is captured at 120 Hz with a 12-camera Vicon motion capture system. The skeletal model of each participant is created with the Vicon BodyBuilder software and markers are attached at predetermined bony landmarks. We simplified the model by ignoring finger joints and extra joints in the spine the Vicon software generates in post processing.

We used two models for pose estimation using the marker data (Fig. 4.1). The Euler angle model uses three orthogonal revolute joints at each 3dof joint of the simplified human skeleton (shoulders, hips, lower back and neck) and uses single revolute joints at hinge joints (elbows, forearms, knees, ankles). To position the model in space, three orthogonal prismatic joints describe the position of the model pelvis relative to the origin, followed by a 3dof Euler angle joint assembly to rotate the pelvis in space. The joint order of the Euler angle model matches the joint order of the skeleton generated by Vicon BodyBuilder. In the Lie groups model, each 3dof rotational joint assembly of the first model is replaced with a SO(3) joint group; hinge joints are represented with SO(2) groups.

Marker position sensors are rigidly attached to each model, then forward kinematics is used to generate the estimated marker positions. Three sets of marker position estimation strategies were used for each data sequence: (1) the Euler angle model and the Vicon IK joint angles, (2) the Euler angle model and a standard EKF algorithm [156], and (3) the Lie group model and the LG-EKF algorithm derived in Sec. 4.2. For each algorithm and each data sequence, pose estimation and marker position estimation was run at multiple data sampling frequencies. The EKF and LG-EKF algorithms were run with an identity covariance matrix, marker observation

Table 4.1 Frequency at which each filtering strategy diverged. The divergence frequency of the LG-EKF is consistently lower than EKF.

Data seq.	B1	B2	D1	D2	R1	R2	So1	So2	St1	St2
EKF [Hz]	**49**	34	22	22	28	22	23	29	**30**	19
LG-EKF [Hz]	**49**	**33**	**21**	**15**	**23**	**21**	**21**	**24**	**30**	**15**

noise of 0.01, and a process noise n value of 300, with parameters empirically determined to provide the best performance for each filter.

4.4.1.1 Pose Estimation Divergence

A model and filtering strategy can be considered more robust if the pose estimation algorithm can still converge at lower sampling frequencies. To evaluate the robustness of EKF and LG-EKF the 10 data sequences were resampled at decreasing frequencies. The mean absolute error (MAE) of each marker position was computed between the original motion capture data (ground truth) and the three estimation strategies. Pose estimation was considered to diverge when the overall MAE of the EKF or LG-EKF data increased above twice that of the Vicon MAE data, and never returned below this threshold. Table 4.1 shows the sampling frequencies at which each filtering strategy diverged for each data sequence.

The filters diverged at lower sampling rates for movements that were less dynamic and/or did not encounter gimbal lock as frequently. EKF pose estimation always diverged at the same or a higher sampling frequency than LG-EKF, and LG-EKF diverged at a significantly lower frequency for movements with slower, more consistent joint motion. The consistently lower divergence frequency of the LG-EKF suggests that the constant acceleration assumption on the group is better for human motion estimation than the constant acceleration assumption in the standard EKF algorithm.

4.4.1.2 Marker Position Mean Absolute Error

Table 4.2 shows the average marker MAE of all data sequences down sampled to 40 Hz, averaged over all data frames. One data sequence diverged when the data was sampled lower than 50 Hz (Table 4.1), so 9 of the 10 data sequences were used in the marker position MAE analyses. Figure 4.8 shows the average MAE of all markers in a sample data sequence. There is a significant difference in the MAE between the Vicon IK estimated marker positions and the two EKF implementations, with the EKF outperforming the Vicon IK solution.

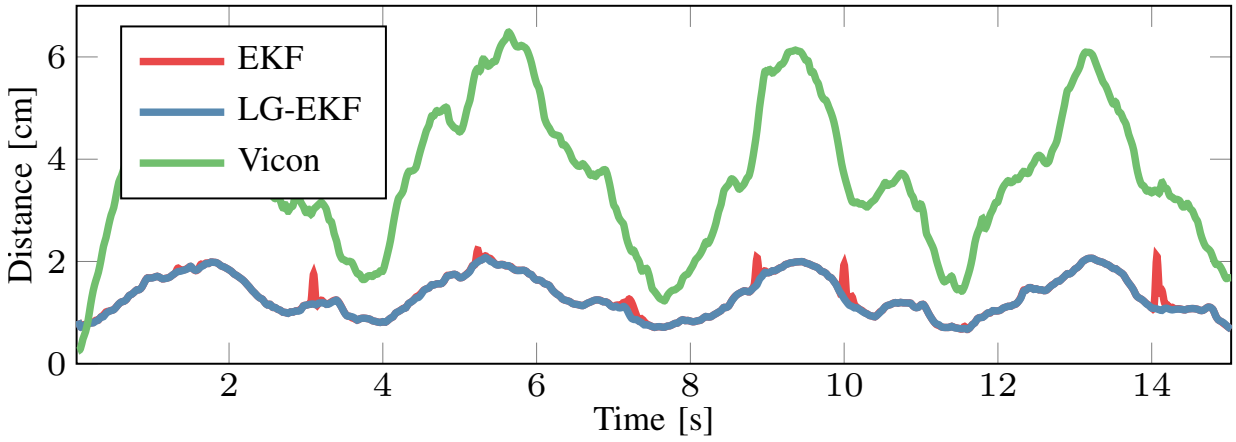


Figure 4.8 Sample of average MAE of all markers for dancing data sequence, down-sampled to 30Hz. Notable spikes in position error are seen when the Euler angle model encounters gimbal lock.

Comparing the two EKF solutions, the primary difference in marker error occurs in regions where an Euler angle 3dof joint encounteres gimbal lock, and during certain highly dynamic motions where the constant acceleration assumption of each filter is severely violated. If a data sequence did not contain very dynamic movement or encounter gimbal lock in the Euler angle model, EKF and LG-EKF results showed negligible differences, especially at higher sampling frequencies.

4.4.1.3 Gimbal Lock Marker Error

We further investigate the estimation accuracy of the two filters when any of the model 3dof rotation joints were within 10% of gimbal lock. Marker error results were generated at sampling rates of 40 Hz, and then averaged over all data sequences and over all frames. The data was down sampled to 40 Hz to emphasize the difference between the two filters.

For all frames containing a 3dof joint within 10% of gimbal lock (second Euler angle within 9° of ± 90°), the marker MAE was tabulated for the EKF, LG-EKF, and Vicon pose estimation methods. The hip joints do not encounter gimbal lock in any of the data sequences. The shoulder joints encounter gimbal lock (when arm is abducted to 90°) in the boxing, dancing and soccer kick data sequences (Table 4.3).

The EKF error is notably higher than LG-EKF error for the right arm within the right shoulder gimbal lock region. Due to the variability of the motions in the dataset, different motions generate gimbal lock in the left and right shoulder. The right shoulder motions involve two axes of rotation,

Table 4.2 Average MAE [mm] of all markers for all data sequences at 40Hz. EKF solutions have similar results, and both outperform the Vicon IK solution. Notation: right (r), left (l), arm (A), leg (L), end-effector (EE).

	Vicon	EKF	LG-EKF
r A	37.25	14.36	**14.18**
l A	34.68	15.00	**14.93**
Torso	23.69	18.60	**18.58**
Head	29.16	17.17	**17.14**
r L	23.93	14.57	**14.44**
l L	25.47	13.07	**12.93**
r A EE	38.20	9.19	**9.00**
l A EE	37.09	9.19	**9.01**
r L EE	20.87	10.04	**9.77**
l L EE	21.96	9.23	**9.08**
All	27.73	14.83	**14.74**

and the LG-EKF significantly outperforms EKF in this case. The left arm error difference in the left shoulder gimbal lock region is not as significant, because the left shoulder motions are primarily in the sagittal plane.

The world-to-pelvis joint encounters gimbal lock in data sequences where the participant rotates their entire body about the global vertical axis, as the second Euler angle was aligned with this axis. Since the torso revolute joint is at the beginning of the kinematic tree it affects the estimation accuracy of the entire body. Table 4.4 (left) shows the MAE of all marker regions when the world-to-pelvis joint is near gimbal lock.

We also compare peak error in the gimbal lock region for all three methods. Table 4.3 (right) shows the arm peak marker error in shoulder gimbal lock regions and Table 4.4 (right) shows all marker regions during the world-to-pelvis joint gimbal lock peaks.

As expected, peak marker errors are greater during gimbal lock as compared to the average error while in the gimbal lock region. The peak marker error with EKF is substantially higher than with LG-EKF at shoulder gimbal lock for both arms, and still greater for each EE. The world-to-pelvis gimbal lock peaks result in an error increase for the entire body, and notably in all end effectors.

Table 4.3 Marker position MAE [mm] during gimbal lock region of shoulders (hip is not shown due to smaller motion range and no gimbal lock). When a shoulder joint encounters gimbal lock the position error on the respective arm of the Euler angle model is significantly greater. This error propagates along the length of the arm and is largest at the end effector (EE). Notation: right (r), left (l), arm (A), leg (L), end-effector (EE).

	Gimbal lock region			Gimbal lock peak		
	Vicon	EKF	LG-EKF	Vicon	EKF	LG-EKF
r A	43.96	31.73	**22.38**	52.53	34.25	**23.95**
r A EE	39.7	29.96	**15.62**	53.75	33.05	**14.68**
l A	44.36	27.9	**26.46**	52.74	33.40	**26.74**
l A EE	45.68	19.42	**18.22**	53.26	33.36	**17.51**

4.4.2 Wearable IMU Dataset

The proposed approach for IMU-based pose estimation was validated through an experiment, where a dynamic figure eight human arm movement was simultaneously recorded with IMUs and motion capture. The IMUs were placed on the humerus and radius. Our IMUs utilize the MPU9250 sensors and sample at 100Hz. Prior to data collection they were calibrated with the method proposed in [157]. To compute their offset and rotation from the humerus and radius, three motion capture markers were placed on each IMU. For the ground truth data and to build a kinematic model of the participant, motion capture markers were placed on the shoulder and medial and lateral sides of the elbow and wrist.

The initial covariance is set to 10^{-3} along the diagonal since in our experiment the initial pose of the participant is known. The IMU observation noise is calculated based on 30 seconds of static data. Assuming constant acceleration process noise [138] of magnitude η, as described in Sec. 4.3, we use the Matlab optimization toolbox to find the optimal process noise parameters for EKF and LG-EKF such that the distance between the estimated and actual elbow and wrist positions is minimized over 3 repetitions of the figure eight motion. The optimal process noise parameters were found to be $\eta_{\text{EKF}} = 389.1$ and $\eta_{\text{LG-EKF}} = 264.8$ for EKF and LG-EKF respectively. The significantly lower optimal process noise for the Lie group motion model shows that human motion is better estimated on the group.

Both filters begin with equally accurate estimation. With each pass near gimbal lock at the corner of the figure eight, EKF accumulates error about the world Z axis. Since LG-EKF is not affected by gimbal lock its performance stays consistent throughout the entire motion. Figure 4.9 shows the estimated and actual wrist positions for both EKF and LG-EKF. Figure 4.10 plots the distance between actual and estimated wrist positions making the error accumulation clear.

Table 4.4 Marker position MAE [mm] during gimbal lock region of world-to-pelvis joint. The error difference between EKF and LG-EKF is notably higher in the EEs as compared to other regions. Notation: right (r), left (l), arm (A), leg (L), end-effector (EE).

	Gimbal lock region			Gimbal lock peak		
	Vicon	EKF	LG-EKF	Vicon	EKF	LG-EKF
r A	52.92	14.13	**13.66**	53.51	16.47	**15.47**
l A	43.15	10.5	**9.98**	42.90	13.25	**11.65**
Torso	41.3	24.25	**24.21**	40.59	25.02	**24.93**
Head	54.62	18.68	**18.48**	58.08	21.67	**20.93**
r L	26.99	13.36	**12.79**	29.00	15.39	**14.10**
l L	26.21	11.09	**10.79**	28.77	13.13	**11.85**
r A EE	53.58	5.39	**4.62**	54.38	8.40	**6.71**
l A EE	44.54	5.15	**4.33**	43.10	8.45	**5.75**
r L EE	19.8	8.15	**7.54**	23.22	10.67	**8.88**
l L EE	28.91	8.26	**7.77**	30.76	10.59	**8.55**
All	36.00	14.01	**13.65**	37.32	16.06	**15.02**

Table 4.5 shows the RMSE and standard deviation for elbow and wrist position estimation.

Table 4.5 Average MAE [mm] of estimated and actual elbow and wrist positions for the two filters. The proposed LG-EKF improves the position estimate by 30% over EKF.

	Elbow RMSE [cm]	Wrist RMSE [cm]
LG-EKF	**5.2 ± 2.6**	**6.9 ± 2.7**
EKF	7.4 ± 3.6	9.9 ± 3.8

Finally we show that group based representation of human motion leads to a better movement variance representation. Both LG-EKF and EKF provide a state error covariance measure which can be used for analyzing human motion. We extract the estimated shoulder position error covariance during the dynamic figure eight motion for both filters and plot its determinant against the second Euler angle estimate which causes gimbal lock (Fig. 4.11). One expects that fast unpredictable motions will have a large error covariance while smooth continuous movements will maintain constant low error covariance. When utilizing the LG-EKF estimator the SO(3) shoulder position error covariance is always presented in the same three perpendicular axes aligned with the upper arm link. However, with EKF error covariance is represented about each of the Euler axes which change with the motion. Because of this, when working with EKF it is difficult to distinguish between gimbal lock and movement related variance sources. At each

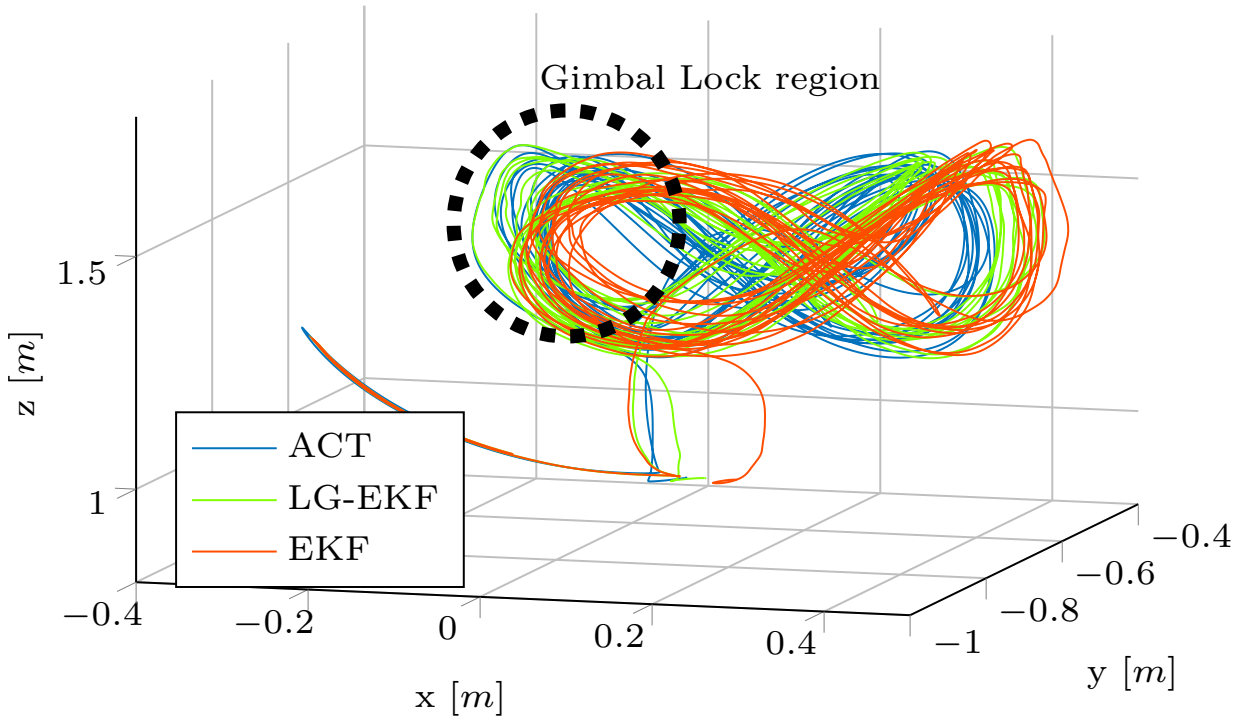


Figure 4.9 Actual and estimated 3d wrist position.

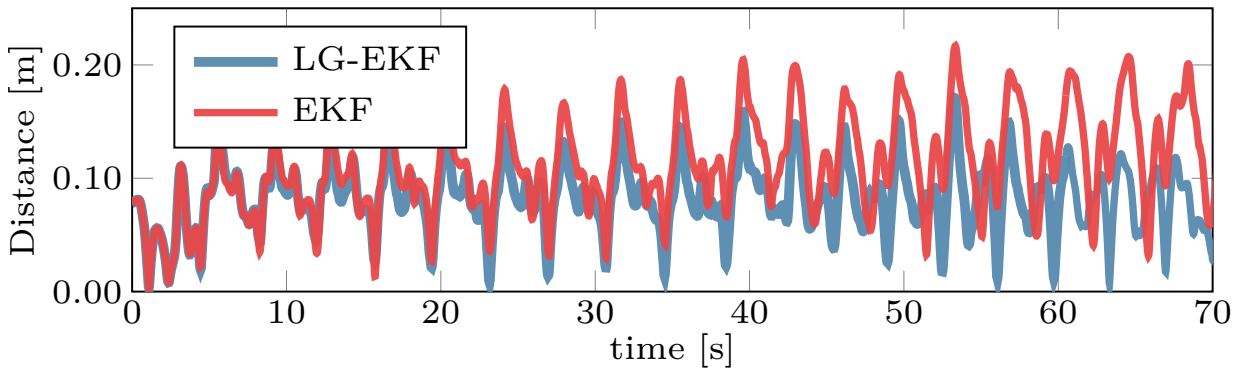


Figure 4.10 Distance between the actual and estimated wrist positions. Euler angle EKF accumulates error about world Z axis with each pass near gimbal lock.

iteration the algorithms also project the error covariance into measurement space, in motion capture applications this can be used to detect missing and incorrectly labeled markers [152]. Since LG-EKF maintains a smooth covariance estimate over smooth motions its projection will be consistent throughout and thus is better suited for such application. Figure 4.12 shows the

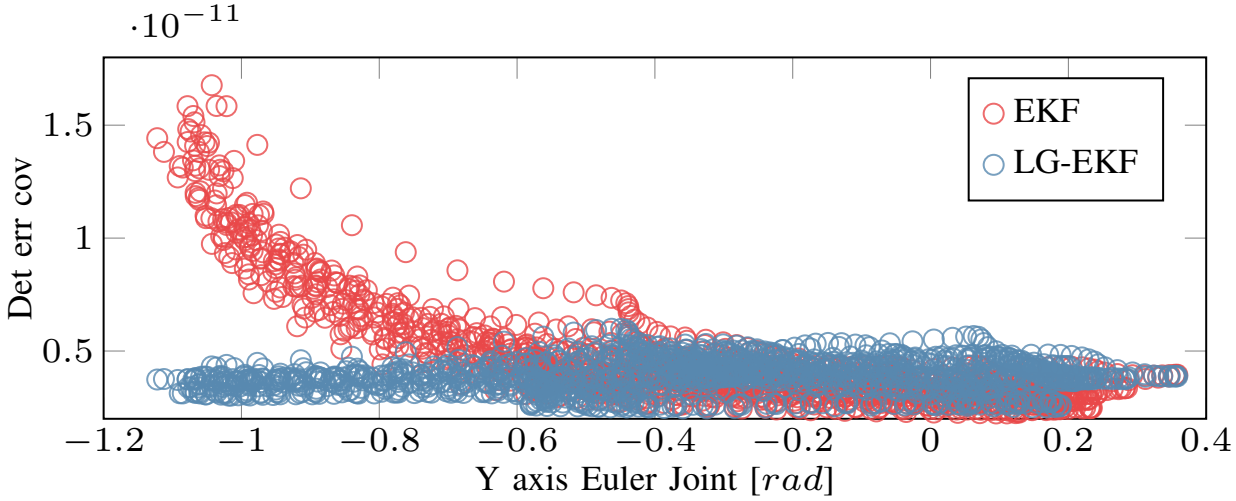


Figure 4.11 Determinant of shoulder position error covariance plotted against the second Euler angle estimate. As EKF approaches gimbal lock at -pi the error covariance increases not due to changes in the motion but due to kinematic modeling. LG-EKF covariance remains smooth through the entire state space.

main eigenvalues of the projected elbow marker covariance during marker based estimation.

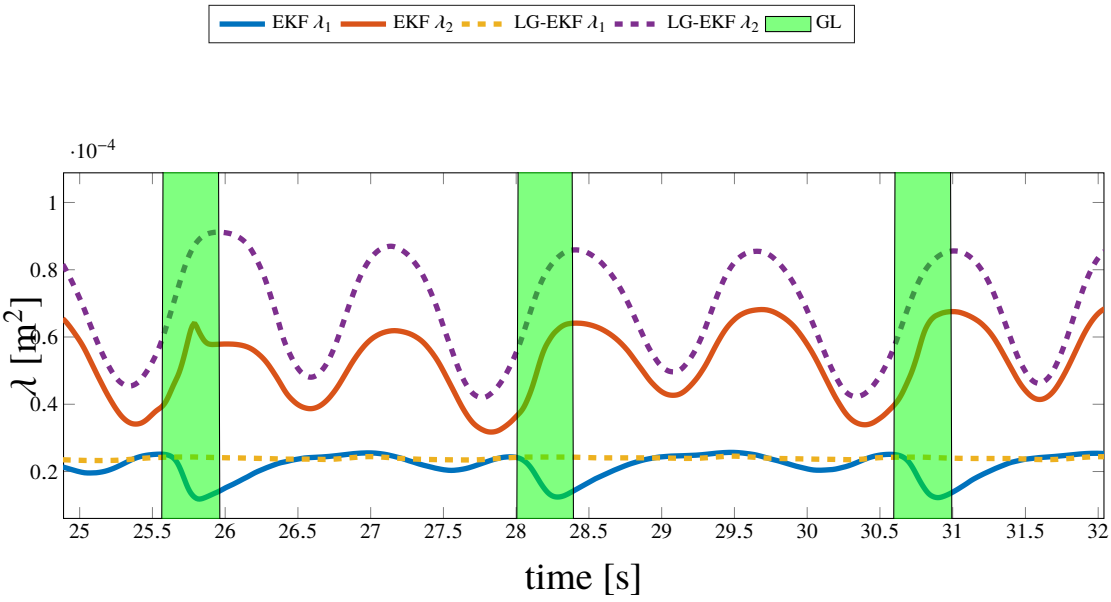


Figure 4.12 Eigenvalues of the error covariance projected to the elbow marker, regions closest to gimbal lock are highlighted. Since the projection is into sensor frame, the first eigenvector is aligned with the direction of motion. Thus we expect variation in the first covariance eigenvalue throughout each motion repetition as the marker experiences directional changes as seen in the LG-EKF plot. However, it is clear that the EKF based estimator covariance is not only effected by the motion but by gimbal lock as well causing error in the covariance estimate in the direction perpendicular to motion.

4.5 Lie or Euler

In the preceding sections we show through simulation and real experiments the benefits of describing human joints as Lie group elements. Lie group based estimation is not affected by gimbal lock, diverges at lower sampling rates, and better captures the error distribution. One drawback of the Lie group formulation in practice is the computational complexity of computing the Jacobians as well as the evaluation of the trigonometric functions present in the $\exp_G$ and $\log_G$ mappings and the iterative calculation of $\Phi_G(\hat{\Omega}_k)$. On the other hand, modeling kinematics with prismatic and revolute joints has been around for a long time and very efficient algorithms have been developed that can easily compute the Euler based Jacobians for systems with hundreds of joints in real time [158, 159]. From figure 4.11 we see that for real wearable sensor data the benefit of utilizing the Lie group shoulder model is only visible when the second Euler joint exceeds -0.6 $rads$ (34 $degrees$). Furthermore, if the movement is aligned with a specific Euler angle there would be no benefit to model the joint on the group. For example, walking is dominated

by flexion and extension of the hip which aligns with the first Euler angle while hip adduction and abduction, which can cause gimbal lock in the Euler model, is below 15 degrees for healthy individuals [160]. Thus when estimating healthy gait using wearable IMUs we would not expect to see a benefit using the Lie group formulation and the Euler model would be preferred due to the reduction in computational complexity. If the movement is known *a-priori*, it is often possible to model a spherical joint using an Euler sequence such that the majority of the motion is aligned with the first angle, avoiding gimbal lock. However, when estimating arbitrary human motion, especially in joints with a large range of motion such as the shoulder, the Lie group formulation will prove advantageous.

4.6 Summary

This chapter proposed a novel algorithm for full body human motion estimation based on motion capture markers and wearable inertial measurement units. The human joints were described as Lie group members, including special orthogonal groups $SO(2)$ and $SO(3)$, and a special euclidean group $SE(3)$. For stochastic inference on Lie groups the LG-EKF was employed, thus explicitly accounting for the non-euclidean geometry of the state space. A constant acceleration motion model for human motion estimation on the group was developed and the Jacobians for markers, gyroscopes, and accelerometers were derived. We show that in order for the motion of an arbitrary kinematic chain to be observable solely based on motion capture markers, a single marker on each link and two at the end effector are required. The proposed approach was extensively validated in both simulation and using real-world motion capture data. Our simulations show that the $SO(3)$ representation of spherical joints is not affected by gimbal lock and provides a consistent error distribution, unlike Euler angle representation, which leads to higher estimation accuracy and easier tuning over regular EKF and UKF. The real world motion capture marker based experiments show that the proposed approach can successfully estimate full body pose at a lower sampling rate than EKF and significantly improves estimation near shoulder and pelvis gimbal lock regions. When estimating human motion using only wearable IMUs the proposed approach significantly outperforms EKF which accumulates more error over time. Furthermore, the estimated $SO(3)$ joint error covariance is dependent only on the motion and not the state thus making it much more appealing for human motion analysis. The chapter also presented the observability analysis of an arbitrarily long kinematic chain of $SO(3)$ elements representing a generalization of a human body. The setup relies on marker position measurements, while the theoretical analysis rests on a differential geometric approach based on Lie derivatives. The conclusions of the observability analysis are also further illustrated through simulations. The framework presented in this chapter does not consider any additional constraints except those

imposed by the kinematic model. Thus the estimated movement would not necessarily satisfy realistic human joint limits or contacts with the environment. Lack of constraints is also why pose estimation using real IMU data required a fixed base model. If we were to model the shoulder as an SE(3) joint, without any position measurement, the accelerometer noise would quickly double integrate into a very large position error. In the next chapter we demonstrate how constraints can be incorporated into the EKF framework to greatly improve the accuracy.

Chapter 5

Constrained human motion estimation

The work presented in this chapter was done in collaboration with Jonathan Lin and Alyson Colpitts from the Department of Electrical and Computer Engineering, University of Waterloo. A portion this chapter has been published in IEEE-RAS 19th International Conference on Humanoid Robots. [161]

In this chapter a framework to incorporate knowledge of constraints into IMU based extended Kalman filter pose estimation is proposed. A novel method to select the most likely currently active constraint from a set is introduced and is applied to show how it can be utilized to accurately estimate the human pose during gait. Unlike many previous methods, described in Section 2.1, that track only the joint angles, the proposed approach is also able to maintain an estimate of the global body position. The proposed approach is extensively validated on real human motion, and the performance is and compared to camera based motion capture.

5.1 EKF Formulation

We switch to utilizing traditional revolute and prismatic joints to model the human body instead of Lie group elements, as described in Section 4.5 it is much more computationally efficient. This allows us to model the human body as a kinematic structure with a tree topology. Each node in the tree represents a revolute or prismatic joint and connections between nodes are the rigid links. To allow arbitrary transformations with respect to the world frame, the root of the tree structure is a chain of 3 revolute and 3 prismatic joints. Each link is assumed to have a rigidly attached IMU. This formulation allows fast computation of forward kinematics and required

Jacobians leading to near realtime performance. A lower body kinematic structure is used for gait estimation (Fig. 5.10). We utilize the EKF to estimate the position, velocity, and acceleration of each joint, including the base-link 6 degree of freedom (DoF) joint, from IMU measurements. Thus, the EKF state consists of the joint positions, velocities, and accelerations of all n joints in the kinematic structure $x = [q_1\ \dot{q}_1\ \ddot{q}_1\ q_2\ \dot{q}_2\ \cdots\ \ddot{q}_n]$. We assume a constant acceleration model $\mathbf{A_i}$ for each joint, resulting in the state update function:

$$\begin{aligned}
\begin{bmatrix} q^i_{k+1} \\ \dot{q}^i_{k+1} \\ \ddot{q}^i_{k+1} \end{bmatrix} &= \underbrace{\begin{bmatrix} 1 & \Delta t & \frac{\Delta t^2}{2} \\ 0 & 1 & \Delta t \\ 0 & 0 & 1 \end{bmatrix}}_{A_i} \begin{bmatrix} q^i_k \\ \dot{q}^i_k \\ \ddot{q}^i_k \end{bmatrix} + \underbrace{\begin{bmatrix} \frac{\Delta t^3}{6} \\ \frac{\Delta t^2}{2} \\ \Delta t \end{bmatrix}}_{w_i} \eta_k \\
x_{k+1} &= \underbrace{\begin{bmatrix} A_1 & 0 & \cdots & 0 \\ 0 & A_2 & \cdots & 0 \\ 0 & 0 & \ddots & \vdots \\ 0 & 0 & \cdots & A_n \end{bmatrix}}_{A} x_k + \underbrace{\begin{bmatrix} w^1 \\ \vdots \\ \vdots \\ w^n \end{bmatrix}}_{w}
\end{aligned} \tag{5.1}$$

where Δt is the sampling interval and η_k represents the acceleration noise during the k-th sample. Process noise w^i is assumed to be zero mean Gaussian (ZMG) with covariance $Q = w^i w^{iT}$. The measurement vector consists of the 3D angular velocity and linear acceleration readings from m wearable IMUs $z_k = [\omega_{1,x}\ \omega_{1,y}\ \omega_{1,z}\ a_{1,x}\ a_{1,y}\ a_{1,z} \cdots a_{m,z}]$. Measurement noise is assumed to be ZMG with covariance $R_{i,j}$ denoting the j-th sensor rigidly attached to the i-th kinematic branch. Forward kinematics can be used to generate a noiseless measurement prediction $\hat{z}_k = h(x_k)$ given the current state [1].

Unfortunately IMU measurements do not provide a global position measurement. Thus, when estimating the transformation of a floating body with respect to the world frame, the gyroscope and accelerometer noise is integrated and double integrated into velocity and position error respectively. To allow for global position estimation we propose to incorporate equality linear constraints on the estimated state, i.e., $C_k \hat{x}_k = b_k$ by projecting the EKF estimate $\hat{x}_k$ onto the constrained space, leading to a constrained state estimate $\hat{x}^P_k$.

$$\hat{x}^P_k = \arg\min_x \{(x - \hat{x}_k)^T W_k (x - \hat{x}_k) : C_k x = b_k\} \tag{5.2}$$

Where W_k is a positive symmetric matrix allowing for different weighting of the state variables. The above minimization has a closed form solution [162]:

$$\hat{x}^P_k = \hat{x}_k - W_k^{-1} C^T (C W_k^{-1} C^T)^{-1} (C_k \hat{x}_k - b_k) \tag{5.3}$$

The constrained error covariance matrix estimate P_k^P can be generated from the original error covariance P_k by using the error covariance definition:

$$P_k^P = \mathbb{E}\left[(x_k - \hat{x}_k^P)(x_k - \hat{x}_k^P)^T\right] \tag{5.4}$$

Substituting the right hand side of Eqn. 5.3 for $\hat{x}_k^P$ into Eqn. 5.4 and noting that expectation is a linear operator, it is possible to re-formulate P_k^P in terms of P_k:

$$P_k^P = (I - W_k^{-1}C^T(CW_k^{-1}C^T)^{-1}C_k)P_k \tag{5.5}$$

To obtain the smallest update to the error covariance, W_k is set to be the unconstrained covariance P_k^{-1} [162]. This maintains the relationship of each joint to its corresponding velocity and acceleration, meaning that the post-projected velocities and accelerations are properly projected into the constrained space as well, even though the constraints may only exist for joint or end-effector positions.

5.2 Constraints

Without an absolute position reference, due to sensor bias integration, the position state estimate will drift away from the true value. Consider estimating human lower body motion during a squat exercise using only wearable IMUs. There are multiple ways to model the lower body; as a free floating kinematic structure with two legs, as a single chain starting as one foot and ending at the other, or as two separate kinematic chains starting at each of the feet. Since no absolute position measurement is available, each model will have associated estimation issues. The free floating lower body will integrate accelerometer noise into velocity and then position, resulting in Cartesian position error for the floating base and the model floating away from the true value. The single chain model is free to integrate error about the world z axis which is not compensated by the accelerometers, this will appear as rotation about the support leg, leading to large position error estimates at the free foot. In the two chain model, the positions of the feet will remain fixed but the drift will cause divergence at the pelvis. The various scenarios are visualized in Fig 5.1. We incorporate prior knowledge into the estimation process as constraints on the state. By selecting what constraints are active as well as what frame they are represented in, complex interactions between the kinematic model and the environment can be incorporated into the pose estimation. Consider the dual kinematic chain lower body described earlier (Fig 5.1, right), by adding a constraint that the pelvis end effector frame of both the left and right leg has the same position and orientation with respect to a global world frame, we can ensure that no divergence occurs. Furthermore, by modelling the constraint of one end effector in the other end effector's frame, we can incorporate complex tasks such as walking while holding onto a rail.

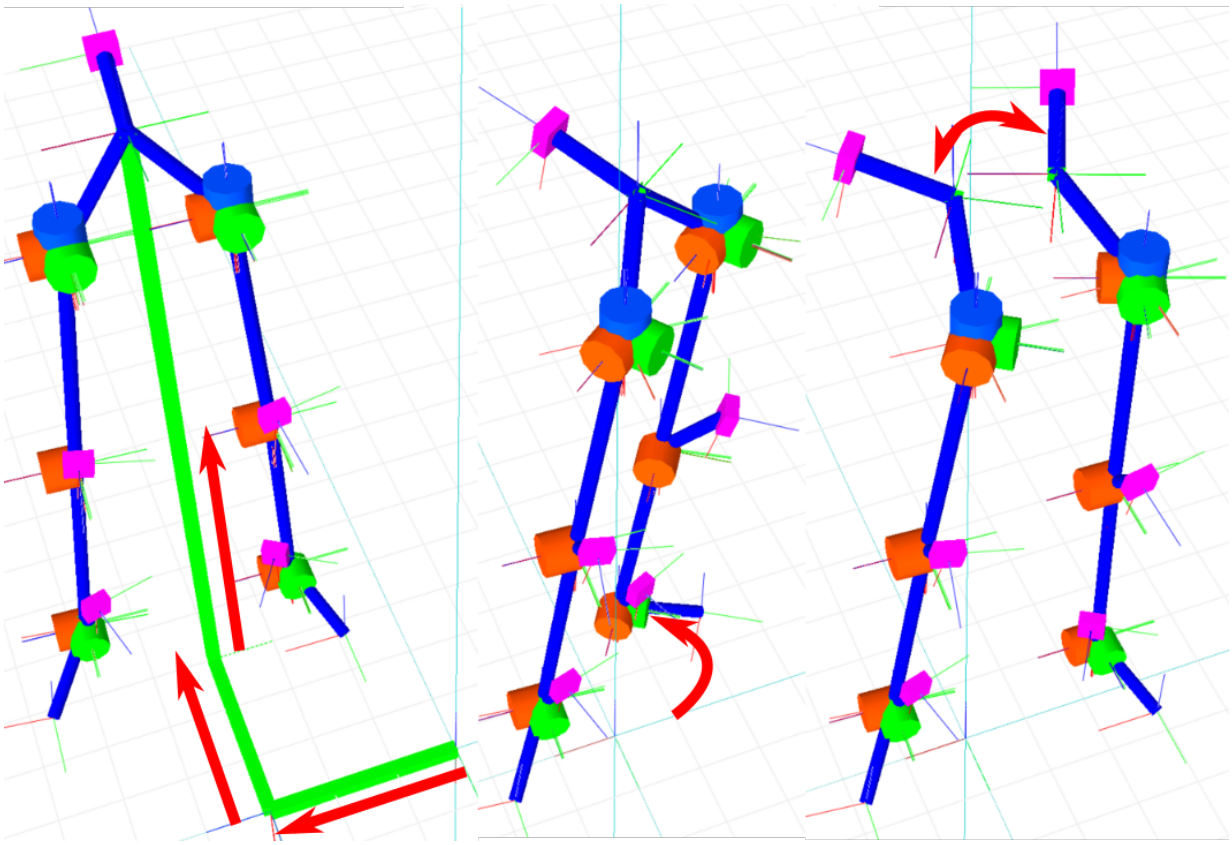


Figure 5.1 Various ways to model the lower body, left to right: floating base, single branch, and two separate branches. Links are denoted by blue rods connecting the revolute joints (red, green, and blue cylinders), pink boxes are wearable IMUs. Based on the modeling method, the error due to sensor noise is accumulated differently (red arrows). The floating base model diverges in global position, the single branch model accumulates drift about the support leg, and the two branch model separates at the pelvis.

.

5.2.1 Base Frame Constraints

Base frame constraints can ensure that a frame in a kinematic structure maintains a desired position and orientation in the world frame. They can also be used to constrain two frames within a kinematic structure in another frame, such as a position of a person's hands in their upper body frame when holding an object with both arms. Consider the forward kinematics of two kinematic branches starting from the same base frame 0, the transformation from the base frame to the frame where they meet must be equal:

$$T_{c1}^{0}(\hat{q}_k^1) = T_{c2}^{0}(\hat{q}_k^2) \tag{5.6}$$

where $\hat{q}_k^1$ and $\hat{q}_k^2$ are the final joint angle estimates for each kinematic branch, respectively. To enforce the constraint, we project the unconstrained state estimate $\hat{x}_k$ after each EKF iteration to the constrained space defined by linearizing the constraint about the predicted EKF state $\hat{x}_k^-$.

Noting that the constrained joint angle estimate is the combination of EKF prediction plus correction $\hat{q}_k^i = \hat{q}_k^{i-} + \Delta q_k^i$ and assuming that the correction is small, we can write out the linearized constraint in the form of $C_k\hat{x}_k = b_k$ (Eq 5.7). Then, by using the minimal rotation representation (roll, pitch, yaw) r and position (x, y, z) p in the base frame, can obtain (Eq 5.8):

$$\begin{aligned}
T_{c1}^{0}(\hat{q}_k^{1-} + \Delta q_k^1) &= T_{c2}^{0}(\hat{q}_k^{2-} + \Delta q_k^2) \\
T_{c1}^{0}(\hat{q}_k^{1-}) + \frac{\partial T_{c1}^{0}(\hat{q}_k^{1-})}{\partial \hat{q}_k^{1-}}(\hat{q}_k^1 - \hat{q}_k^{1-}) &= T_{c2}^{0}(\hat{q}_k^{2-}) + \frac{\partial T_{c2}^{0}(\hat{q}_k^{2-})}{\partial \hat{q}_k^{2-}}(\hat{q}_k^2 - \hat{q}_k^{2-}) \\
\underbrace{\left[\frac{\partial T_{c1}^{0}(\hat{q}_k^{1-})}{\partial \hat{q}_k^{1-}},\ -\frac{\partial T_{c2}^{0}(\hat{q}_k^{2-})}{\partial \hat{q}_k^{2-}}\right]}_{C} \underbrace{\begin{bmatrix}\hat{q}_k^1 \\ \hat{q}_k^2\end{bmatrix}}_{x} &= \underbrace{\left[\frac{\partial T_{c1}^{0}(\hat{q}_k^{1-})}{\partial \hat{q}_k^{1-}},\ -\frac{\partial T_{c2}^{0}(\hat{q}_k^{2-})}{\partial \hat{q}_k^{2-}}\right]\begin{bmatrix}\hat{q}_k^{1-} \\ \hat{q}_k^{2-}\end{bmatrix} + T_{c2}^{0}(\hat{q}_k^{2-}) - T_{c1}^{0}(\hat{q}_k^{1-})}_{b} \qquad (5.7) \\
\left[J^b(\hat{q}_k^{1-}),\ -J^b(\hat{q}_k^{2-})\right]\begin{bmatrix}\hat{q}_k^1 \\ \hat{q}_k^2\end{bmatrix} &= \left[J^b(\hat{q}_k^{1-}),\ -J^b(\hat{q}_k^{2-})\right]\begin{bmatrix}\hat{q}_k^{1-} \\ \hat{q}_k^{2-}\end{bmatrix} + \begin{bmatrix}r \\ p\end{bmatrix}(\hat{q}_k^{2-}) - \begin{bmatrix}r \\ p\end{bmatrix}(\hat{q}_k^{1-}) \qquad (5.8)
\end{aligned}$$

where $J^b(\hat{q}_k^{i-})$ is the base frame Jacobian evaluated at the predicted state. Lastly, the above formulation may have problems due to different rotation representations in roll, pitch, yaw, resulting in incorrect differences of the angles. Under the small angle assumption we can approximate $r(\hat{q}_k^{1-}) - r(\hat{q}_k^{2-})$ as (Eq 5.9):

$$r(\hat{q}_k^{1-}) - r(\hat{q}_k^{2-}) \approx rpy(R_{c1}^{0}R_{0}^{c2}) \tag{5.9}$$

where $rpy(R)$ is the roll, pitch, yaw representation of a rotation matrix.

5.2.2 End Effector Frame Constraints

Apart from constraining two frames in a common base frame it can be desirable to constrain one frame with respect to the other. For example, when holding a rail, the hand is only free to slide along a specific axis in the rail's frame. Linearizing the transformation from one end effector to the other and using the row of the Jacobian corresponding to the sliding direction allows to enforce such a constraint.

Position First we consider the position $t = [t_x\ t_y\ t_z]$ of one end effector in the frame of another (Eq 5.10):

$$t = T_{c2}^{c1}(\hat{q}_k^1, \hat{q}_k^2) \cdot 0 = T_0^{c1}(\hat{q}_k^1) T_{c2}^0(\hat{q}_k^2) \cdot 0 \tag{5.10}$$

where $0 = [0, 0, 0, 1]^T$ is a zero homogeneous coordinate vector representing the local frame origin. Linearizing as before we obtain a linear constraint on the position in terms of the estimated state (Eq 5.11):

$$\underbrace{\left[\frac{\partial T_0^{c1}(\hat{q}_k^{1-})}{\partial \hat{q}_k^{1-}} T_{c2}^0 + T_0^{c2} \frac{\partial T_{c2}^0(\hat{q}_k^{2-})}{\partial \hat{q}_k^{2-}}\right]}_{C} \underbrace{\begin{bmatrix}\hat{q}_k^1 \\ \hat{q}_k^2\end{bmatrix}}_{x} = \underbrace{\left[\frac{\partial T_0^{c1}(\hat{q}_k^{1-})}{\partial \hat{q}_k^{1-}} T_{c2}^0 + T_0^{c2} \frac{\partial T_{c2}^0(\hat{q}_k^{2-})}{\partial \hat{q}_k^{2-}}\right] \begin{bmatrix}\hat{q}_k^{1-} \\ \hat{q}_k^{2-}\end{bmatrix} - T_{c2}^{c1}(\hat{q}_k^1, \hat{q}_k^2) \cdot 0}_{b} \tag{5.11}$$

The position constraint is then controlled by selecting the appropriate rows. For example, when holding a rail that extends in the x axis of the rail frame, the y and z position of the hand would be constrained to zero in the rail frame.

Rotation Finally, by directly considering the rotation matrix from two end effectors to a common frame, it is possible to align the desired end effector axes. Consider holding a bar with both hands, while each hand is free to rotate around the bar independently, the axis along the bar must be equivalent for both. Linearizing the rotation matrix from the end effectors to a common base frame and selecting the desired common axis allows us to incorporate this constraint into the estimation (Eq 5.12):

$$\underbrace{\left[\frac{\partial R_{c1}^0(\hat{q}_k^{1-})_l}{\partial \hat{q}_k^{1-}}, -\frac{\partial R_{c2}^0(\hat{q}_k^{2-})_l}{\partial \hat{q}_k^{2-}}\right]}_{C} \underbrace{\begin{bmatrix}\hat{q}_k^1 \\ \hat{q}_k^2\end{bmatrix}}_{x} = \underbrace{\left[\frac{\partial R_{c1}^0(\hat{q}_k^{1-})_l}{\partial \hat{q}_k^{1-}}, -\frac{\partial R_{c2}^0(\hat{q}_k^{2-})_l}{\partial \hat{q}_k^{2-}}\right] \begin{bmatrix}\hat{q}_k^{1-} \\ \hat{q}_k^{2-}\end{bmatrix} + R_{c2}^0(\hat{q}_k^{2-})_l - R_{c1}^0(\hat{q}_k^{1-})_l}_{b} \tag{5.12}$$

where l denotes the desired column of the rotation matrix representing an axis about which the constraint is enforced.

It is important to note that both base and end effector frame constraints are six dimensional and can be broken down into their 3 dimensional position and orientation components. For example, while holding a metal railing one can still slide and rotate their hand along and about the railing. Thus, only four of the six constraints are enforced. To apply multiple constraints simultaneously, the C matrices and b vectors of each individual constraint can be vertically stacked.

5.3 Active Constraint Detection

Incorporating constraints can significantly improve pose estimation of kinematic structures. However, while some constraints remain constant, other constraints are only active for some time. Determining the active constraint in real time can greatly expand the usability of the method. To choose one active constraints out of a possible set we project the unconstrained state estimate to each constrained subset and predict the resulting constrained measurement vectors. Comparing the weighted norm of these constrained measurement vectors allows us to select the most likely active constraint.

Consider m possible constraints $C_k^j x = b_k^j | j = 1 : m$ we would like to determine which constraint is active at time k and should be included in the estimation approach. For each possible constraint we project the unconstrained state $\hat{x}_k$ and covariance P_k onto the constrained subset.

$$\hat{x}_k^{Pj} = \hat{x}_k - W_k^{-1} C^{jT} (C^j W_k^{-1} C^{jT})^{-1} (C_k^j \hat{x}_k - b_k) \tag{5.13}$$

$$P_k^{Pj} = (I - W_k^{-1} C^{jT} (C^j W_k^{-1} C^{jT})^{-1} C_k^j) P_k \tag{5.14}$$

After the projection we can predict the each constrained measurement residual and its covariance using the observation model.

$$\tilde{z}_k^{Pj} = z_k - h(\hat{x}_k^{Pj}) \tag{5.15}$$

$$S_k^{Pj} = H_k^{Pj} P_k^{Pj} H_k^{PjT} \tag{5.16}$$

where $H_k^{Pj} = \frac{\delta h(x)}{\delta x}\Big|_{x_k^{Pj}}$ is the observation model Jacobian evaluated at the projected state. We now have a weighted metric of how well applying each constraint matches the real measurements

$$d_k^j = \tilde{z}_k^{PjT} S_k^{Pj} \tilde{z}_k^{Pj} \tag{5.17}$$

A naive approach for selecting the most likely active constrain is to take j corresponding to the minimum d_k^j, however, the measurement noise in z_k can lead to incorrect selection and rapid switching of the active constraints. To overcome this, we propose a look ahead constrained Kalman filter method where a horizon of projected measurement residuals is used to determine the active constraint.

We begin by running separate EKFs for each constraint for a horizon of H steps. The active constraint j_{act} with the lowest metric sum over the horizon and the respective state $\hat{x}_1^{Pj_{act}}$ and covariance $P_1^{Pj_{act}}$ are then the result of the estimation at the first time-step.

$$j_{act} = \min_i \sum_{k=1}^{H} d_k^i \tag{5.18}$$

Next at each iteration the active contraint EKF performs a single iteration to propagate the horizon one step forward. The non-active constraints EKFs are reset to the current active state $\hat{x}_k^{Pj_{act}}$ and covariance $P_k^{Pj_{act}}$ and run for the entire horizon computing the metric sum. Finally, the metric sums are compared and the active constraint is re-selected.

The proposed approach requires $(m-1)H+1$ EKF iterations per time-step to build the cumulative metric sums for all of the possible constraints. Of course this is computationally expensive and it is difficult to run the algorithm in real-time without parallel computing since each EKF iteration inverts a $3n \times 3n$ matrix. We can greatly reduce the number of required iterations by noting that human motion is continuous and thus for constraint j to become active, the projected state estimate $\hat{x}_k^{Pj}$ has to be close to the current unconstrained state estimate $\hat{x}_k$. Thus, the metric sum over the horizon is only computed for those constraints where $\|\hat{x}_k - \hat{x}_k^{Pj}\| < \alpha$ where α is a tuning parameter. The entirety of each iteration of the active constraint detection algorithm is as follows:

Algorithm 1 Active constraint selection at each time step

Result: $\hat{x}_k^{Pj_{act}}$ and $P_k^{Pj_{act}}$
Compute unconstrained estimate $\hat{x}_k$
for *constraints* $j = 1 : m$ **do**
  Compute x_k^{Pj} using (5.13)
  if $\|\hat{x}_k - \hat{x}_k^{Pj}\| < \alpha$ **then**
    Run Constrained EKF for horizon of H
    Compute $d_{k:k+H}^{j}$ using (5.17)
  end
end
Choose active constraint j_{act} using (5.18)

Thus the proposed approach is capable enforcing constantly active constraints as well as selecting the most likely active constraint from a set.

5.4 Application for Gait estimation

To better illustrate the active constraint detection method and how the proposed approach can be applied for practical pose estimation we begin with a simplified simulation of walking. Consider the gait of a planar biped containing one 2-DoF prismatic and two 1-DoF revolute joints (Fig. 5.2) with 3 IMUs attached at the feet and pelvis. The two prismatic joints describe the translation of the robot's base with respect to the world frame. Assuming that the revolute joints follow a sinusoidal trajectory and the support foot is switched to the leading leg at the joint angle maximum, we can find a closed form solution to the world x and z axes aligned prismatic joint positions ($p_x,\ p_z$) such that the support foot is static.

$$\theta = \theta_{max} sin(t) \tag{5.19}$$
$$\omega = \theta_{max} cos(t) \tag{5.20}$$
$$\dot{p}_x = \|Lcos(\theta)\omega\| \tag{5.21}$$
$$p_z = Lcos(\theta) \tag{5.22}$$

where L is the leg length, $\theta_{max} = 20^\circ$ represents the revolute joint maximum angle and ω is the revolute joint angular velocity. Differentiating and integrating the above allows us to compute the

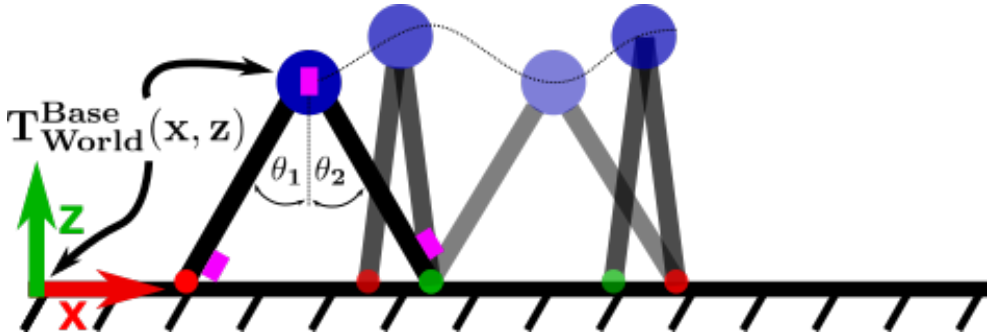


Figure 5.2 Planar biped gait simulation. Two prismatic joints connect the biped's body to the world frame and correspond to the x and z positions of the base. Each leg is connected with a revolute joint (θ_1, θ_2) and experiences the same sinusoidal motion. The support foot is switched at the joint angle maxima. The body of the biped experiences a trajectory similar to an inverted pendulum during each step. Simulated IMUs (purple rectangles) are attached at the pelvis and feet.

positions, velocities, and accelerations of all 4 joints in the biped. Utilizing forward kinematics IMUs are simulated sampling at 100 Hz ($\Delta t = 0.01$). ZMG noise with standard deviation of 0.1 and 0.01 is added to the accelerometer and gyroscope measurements respectively (Fig. 5.3).

We formulate a single 2-DoF parent frame constraint in the world x and z axis to fix the stance foot to the ground. The proposed approach is used to select the currently active constraint and estimate the biped pose. At the end of each iteration the inactive constraint (belonging to the swing foot) is updated by projecting the estimated foot position to the ground plane. This effectively slides the swing foot constraint on the ground plane until it is selected as active, allowing the biped to walk in the x direction.

Assuming that process noise is only present in acceleration, w^i is set to $[\frac{1}{6}\Delta t^3 \ \frac{1}{2}\Delta t^2 \ \Delta t]\eta$ where for this simulation $\eta = 10^4$ is a the process noise tuning constant. The measurement noise R is set to 0.1 and 0.01 for the accelerometer and gyroscope respectively. The look ahead horizon is 10 samples and to ensure that the metric d_k is always computed for both constraints, for this simulation, $\alpha = \infty$. Fig. 5.4 shows the actual and estimated prismatic and revolute joint positions as well as the constraint switches. The approach correctly maintains one of the feet always fixed to the ground by accurately identifying the active constraint. This prevents error accumulation in the prismatic joints and provides precise revolute joint and global position tracking of the biped. The proposed metric to detect the currently active constraint is shown in Fig. 5.5. With the swing foot constrained to the ground plane the biped is unable to produce motion to match the foot IMU measurements, thus the metric increases. For the correct active constraint the metric closely follows the magnitude of the measurement noise. Finally, Fig. 5.6 shows how the α parameter can be used to avoid running horizon estimation at every time step. When the the

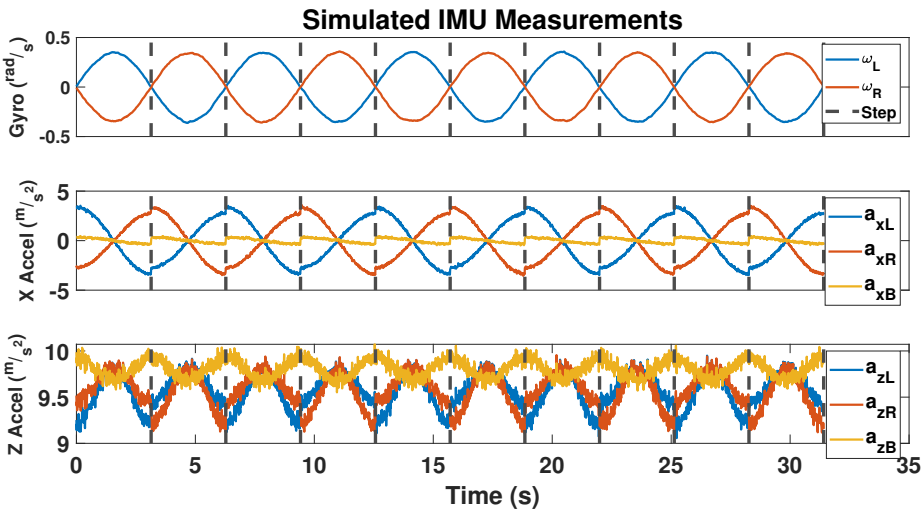


Figure 5.3 Planar biped simulated IMU measurements during gait for the sensors attached at the left (L) and right (R) feet as well as the body (B). Due to the planar motion, acceleration (a) is only present in the x and z axes. Furthermore, since the body connects to the world frame through 2 prismatic joints it can only experience linear velocities and thus only the angular velocities ω about the y axis (perpendicular to the planar motion) at the feet are considered. Black vertical dashed lines indicate the support switching points.

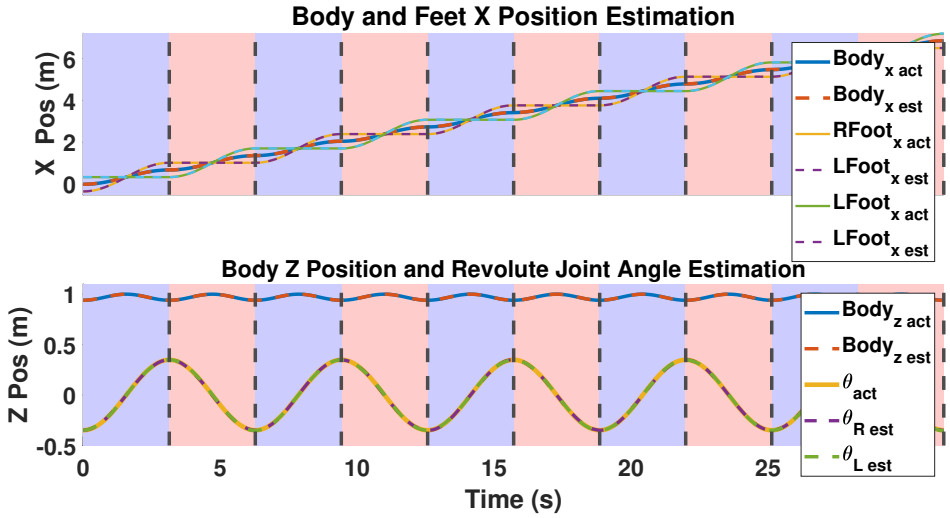


Figure 5.4 Biped gait pose estimation. The proposed approach accurately estimates motion in both prismatic and revolute joints and correctly detects support foot switches. Top: x axis estimated and actual position of the body and feet. The mean absolute tracking errors (MAE) are 0.35 cm and 0.32 cm for the feet and body respectively. Bottom: Z axis body position (0.027 cm MAE) and revolute joint angles tracking (0.075° MAE). Shaded red and blue regions correspond to the estimated right and left support foot respectively. Black vertical dashed lines indicate the true support switches.

distance between projected and unconstrained state estimates is large it cannot become active as that would imply discontinuous motion. In this case we do not need to estimate the error over the look ahead horizon and only one EKF iteration is required. Notice that α can easily be tuned by plotting the distance between the constrained and unconstrained estimates and then setting it sufficiently high so all the constraint switches happen below the α threshold.

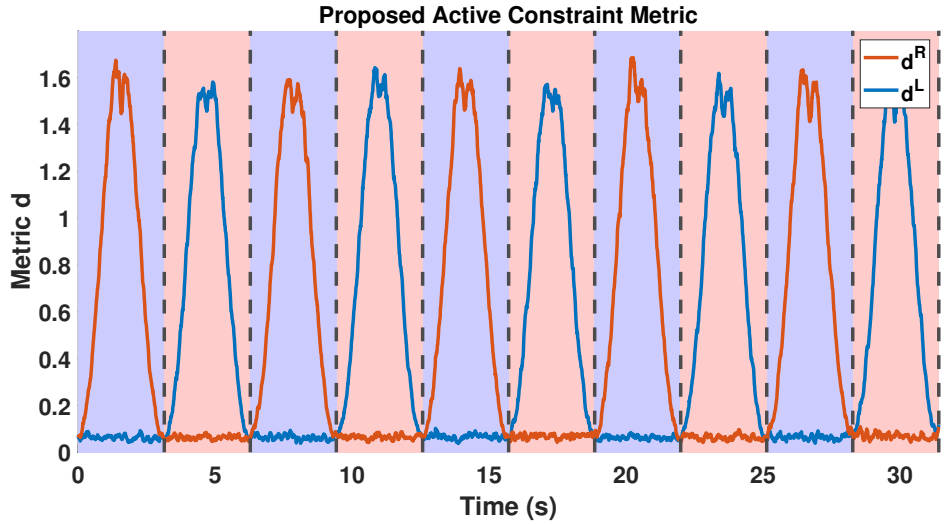


Figure 5.5 Proposed active constraint detection metric d for the left (d^L) and right (d^R) foot. A support switch is detected when the minimum metric changes. In the worst case scenario the switch detection was offset by 4 samples (0.04 seconds).

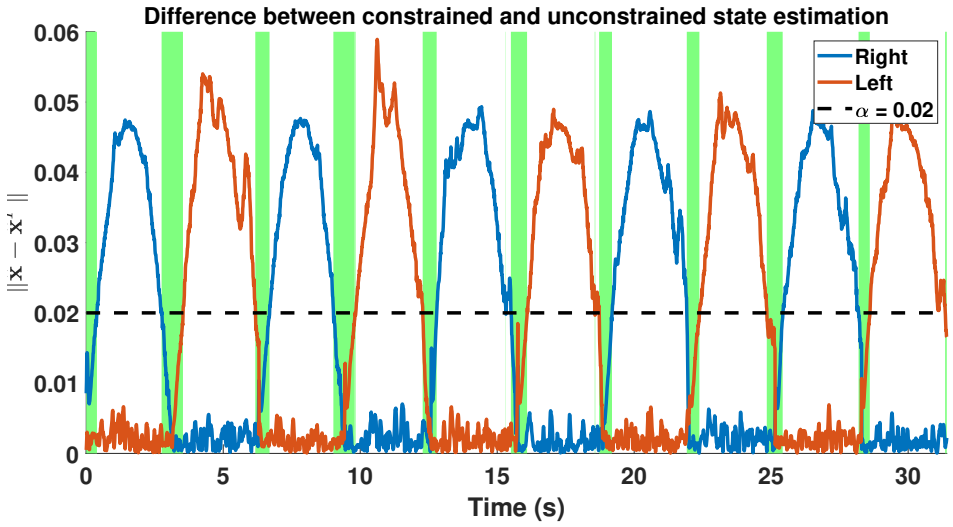


Figure 5.6 Distance between the constrained and unconstrained state estimates of the left and right feet. Green shaded regions indicate the distance less than $\alpha = 0.02$ for either foot and represent 18% of the data. Outside of those regions only a single EKF iteration is necessary.

5.5 Experiments

Extensive validation with real human motion has been performed to both test the various proposed linearized constraints as well as the active constraint detection method. Ground truth joint angle and joint centre trajectories from motion capture were calculated using a marker-based EKF estimator [163]. The algorithm is based on an unconstrained EKF that utilizes very accurate motion capture marker positions as measurements instead of IMUs, it also automatically detects and handles missing or swapping markers. For comparison, a non-constrained IMU-based EKF estimator was used [1].

5.5.1 Constant Constraints

The proposed base frame and end effector frame constraints were tested on two human motion datasets collected with wearable sensors developed in our lab. The first set of data is a 29 healthy participant (15 F, 14 M) exercise set collected from the student population at the University of Waterloo. Participants performed 10 to 20 repetitions of squats given only verbal instructions. The IMUs were placed on the right and left calf, thigh, and the torso using double-sided medical adhesives (Figure 5.7). For the proposed method, the right and left legs were modelled as two separate kinematic chains fixed at each foot, starting from the ankle joint and ending at the pelvis centre, with position and orientation constraints in all three axes locking together the right version of the pelvis centre to the left. For comparison, 3 other unconstrained EKF configurations were considered: (1) a floating base at the pelvis centre with the two legs branching from the base, (2) a fixed base at the pelvis centre with the two legs branching from the base [1], and (3) a fixed base at the right foot, with the kinematic chain extending through the hips and pelvis, to the left foot [1, 164]. Only 23 of the 29 participants were processed due to missing markers causing erroneous link length estimation or sensor placement, or errors in the time alignment between IMU and motion capture data.

In the second dataset, 2 participants (2 M) performed 10 to 20 repetitions of upper body closed chain tasks with a metal bar. The IMUs were placed on the right and left upper arm, forearm, and the hands using double-sided medical adhesives (Fig 5.8). The exercises performed were the bench press, rotating the bar back and forth while maintaining two-handed contact, and sliding the right hand along the bar while still holding it with both hands. The kinematic model for the proposed method was a fixed base at the right and left shoulder, with constraints from the left to the right hand. For the bench press and the bar rotation exercise, the right hand was constrained in both position and orientation in all three axes relative to the left hand. For the sliding task, the right hand was allowed to change in the local y-direction (Fig 5.9), allowing its distance to

Figure 5.7 The placement of torso, knee, and ankle IMUs for the squat exercise is shown here, as well as the locations of the motion capture markers for ground truth joint angle trajectory estimation.

the left hand to change along the bar, but not in the other two directions, which prevents the right hand from leaving the bar in the model. The kinematic model for the comparison EKF was identical but without the left to right hand constraint [1]. Motions where hands did not maintain constrained contact with the bar were excluded.

Using the IMU accelerometer and gyroscope data, joint angles and joint centre trajectories were estimated using the proposed technique. To provide ground truth data for comparison, the joint angles and joint centre trajectories from the motion capture data were separately estimated using the approach described in [163]. The link lengths and sensor locations for all models were calculated from the motion capture data, and the hip rotation centre was calculated using

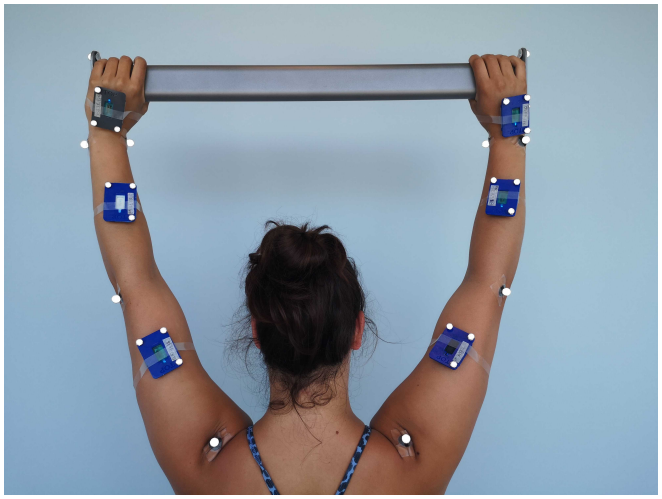


Figure 5.8 The placement of the upper arm, lower arm, and hand IMUs for the upper body metal bar exercises is shown here, as well as the location of the motion capture markers for ground truth joint angle trajectory estimation.

anthropometrics [165]. The IMU and motion capture data were aligned using a sync motion at the start and end of the motion. For each exercise type, the same Kalman process and observation noise covariances were used for both the proposed method and all comparison Kalman filters. Joint angle error was reported as rotation matrix similarity [166] so that alternative solutions in the spherical joints can be compared directly, instead of relying on the Euclidean distance in the cases where the spherical joint resolved to its Euler angles differently between models. This metric m calculates the similarity between two rotation matrices by taking the Frobenius norm of the form (Eq 5.23):

$$\begin{aligned} R_{RMS} &= R_{imu} R_{mocap}^{T} \\ m &= ||I_3 - R_{RMS}||_F \end{aligned} \tag{5.23}$$

where I_3 is the identity matrix, and R_{imu} and R_{mocap} are the two matrices being compared. The resulting metric is normalized to have a range from 0 to 1, where 1 denotes perfect similarity between the two rotation matrices, and 0 denotes a rotation of 180 degrees. Although this metric allows both spherical joints, such as the hip and shoulder, and single revolute joints, such as the knee and elbow, to be assessed using the same approach, it is important to note that the spherical joints consist of three degrees of freedom, thus the similarity metric provides a summary view of the error across the 3 DoF, while the same metric corresponds to the joint angle error for the 1 DoF joints. The similarity metric can also be converted into a degree representation between 0

and 180 degree error about its rotation axis for a more intuitive interpretation (Eq 5.24):

$$\theta = \arccos\left(tr(R_{RMS}) - 1\right)/2 \tag{5.24}$$

For joint position error, the root-mean-square of the Euclidean norm was used.

Table 5.1 Joint angle rotation matrix similarity [0 to 1, where 0 is no error], joint angle error [deg], and joint position error [cm] for the squatting dataset. The best performing model in each metric is bolded. Note that the hip and ankle joint angle similarity error consists of 3 individual joints while the knee joint consist only a single joint.

Metric	Model	Average	R Hip	R Knee	R Ankle	L Hip	L Knee	L Ankle
Angle similarity [0:1]	Floating		0.42	0.74	0.86	0.44	0.75	0.87
	Fixed pelvis		0.71	0.95	0.86	0.70	0.95	0.87
	Fixed R foot		0.79	**0.96**	0.90	0.77	**0.96**	0.87
	Proposed		**0.84**	0.95	**0.91**	**0.84**	0.95	**0.92**
Joint angle error [deg]	Floating		70.90	30.14	16.10	68.11	28.96	14.94
	Fixed pelvis		33.72	5.73	16.10	34.92	5.73	14.94
	Fixed R foot		24.24	**4.58**	11.48	26.59	**4.58**	14.94
	Proposed		**18.41**	5.73	**10.33**	**18.41**	5.73	**9.18**
Position error [cm]	Fixed R foot	7.17	6.14	**4.16**	**1.39**	9.49	11.63	9.96
	Proposed	**3.46**	**5.45**	4.35	**1.39**	**5.61**	**4.88**	**1.38**

Table 5.1 shows the the joint angle rotation matrix similarity and the joint position error compared to motion capture estimates for the floating base pelvis, fixed pelvis, fixed foot, and the proposed method on the squatting dataset. The floating base and fixed pelvis models lead to position errors exceeding 20 cm and are excluded from this table. For the floating base case, since the IMUs do not provide any absolute position data, EKF is able to satisfy the sensor measurements with movements in the floating base, leading the estimated model to drift off towards infinity, reaching an average position error of over 1300 cm. For the fixed pelvis model, as the pelvis centre was the start of the kinematic chain and was fixed in space, the participant is effectively jumping up instead of squatting to the ground, which leads to a mismatch between the direction of movement between the motion capture markers and the kinematic model of the fixed pelvis inverse kinematics.

The proposed approach was demonstrated to outperform the 3 comparison methods due the proposed constraints attenuating the impact of drift introduced by sensor bias and sensors shifting due to muscle motions, with the proposed method achieving 3.5 cm of average position error, while the fixed foot approach achieves 7.2 cm of position error. Table 5.1 shows that in many

cases individual joint angle recovery of the different models do not deviate strongly from the ground truth, and that properly anchored kinematic chains with EKF are comparably effective at recovering the 1 degree of freedom (DOF) knee joint, with error under 6 degrees compared to the motion capture ground truth. The effects of drift are more observable in the 3 DOF hip and ankle spherical joints, where the proposed method was shown to be up to 26 degrees closer to the ground truth data. The accumulated effect of several inaccurate joints leads to a high position error at the end of the kinematic chain, while the correction due to the proposed constraints results in a significantly smaller left ankle position error, where the proposed method achieved just over 1 cm error, while the fixed foot model obtained an error of 10 cm. The smaller left ankle error compared to the knee and foot is due to some instances of low right hip drift in the internal rotation direction and high left hip drift in the same direction, leading the left knee to point away from the direction of the squat but still allowing the left ankle to remain relatively close to the ground truth position.

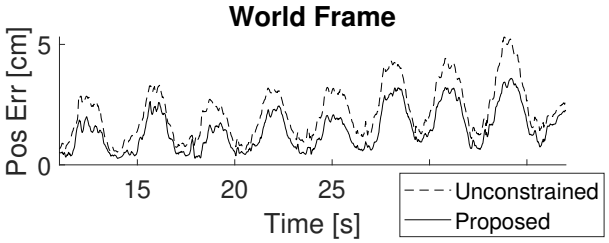


Figure 5.9 The average right hand position error with respect to motion capture in the world frame of the hand sliding along a bar task is shown. The effects of this drift can be seen in the world frame position error in the bottom figure, growing without bound.

For the bench press dataset, Fig 5.9 shows a comparison of the position error between the unconstrained EKF and the proposed method. Table 5.2 shows the overall joint angle similarity and position error, showing an average joint position error of 2.9 cm while the comparison unconstrained EKF method shows an average joint position error of 4.4 cm. Similar to Table 5.1, the bench press tests show that while individual joint angles may be estimated well, the errors accumulate to large joint position errors in the wrists, where the proposed method outperforms the unconstrained EKF by up to 6 cm. However, the proposed method is sensitive to violation of the constraint. In the sliding task, the right wrist is shown to perform poorly compared to the unconstrained case in the joint angle similarity due to minor hand twisting around the bar, causing the proposed method to attribute the motion as wrist movement when in fact, relative to the hand, the bar rotated. Since the unconstrained case does not have knowledge of the bar, it is able to outperform the proposed method in these cases.

Table 5.2 Joint angle rotation matrix similarity [0 to 1, where 0 is no error], joint angle error [deg], and joint position error [cm] for the bench press dataset. The best performing model in each metric is bolded. Note that the shoulder and hand joint angle similarity error consists of 3 individual joints while the elbow joint consist only a single joint.

Metric	Exercise	Model	Average	R Shoulder	R Elbow	R Wrist	L Shoulder	L Elbow	L Wrist
Angle similarity [0:1]	Bench	Unconstrained		0.83	**0.93**	**0.92**	0.75	**0.92**	**0.91**
		Proposed		**0.92**	**0.93**	**0.92**	**0.91**	**0.92**	0.90
	Rotation	Unconstrained		0.92	**0.96**	0.95	**0.92**	**0.95**	**0.95**
		Proposed		**0.94**	0.93	**0.96**	0.90	0.92	**0.95**
	Slide	Unconstrained		0.93	0.92	**0.96**	0.93	0.90	**0.96**
		Proposed		**0.94**	**0.93**	0.85	**0.94**	**0.95**	**0.96**
Joint angle error [deg]	Bench	Unconstrained		19.58	**8.03**	**9.18**	28.96	**9.18**	**10.33**
		Proposed		**9.18**	**8.03**	**9.18**	**10.33**	**9.18**	11.48
	Rotation	Unconstrained		9.18	**4.58**	5.73	**9.18**	**5.73**	**5.73**
		Proposed		**6.88**	8.03	**4.58**	11.48	9.18	**5.73**
	Slide	Unconstrained		8.03	9.18	**4.58**	8.03	11.48	**4.58**
		Proposed		**6.88**	**8.03**	17.25	**6.88**	**5.73**	**4.58**
Position error [cm]	Bench	Unconstrained	6.73	**0.00**	8.85	8.72	**0.00**	11.70	10.18
		Proposed	**3.45**	**0.00**	**3.95**	**4.67**	**0.00**	**4.43**	**5.63**
	Rotation	Unconstrained	3.66	**0.00**	4.61	5.67	**0.00**	**3.53**	**4.25**
		Proposed	**2.83**	**0.00**	**2.75**	**3.21**	**0.00**	3.86	4.47
	Slide	Unconstrained	2.88	**0.00**	3.40	3.42	**0.00**	3.37	3.84
		Proposed	**2.36**	**0.00**	**2.83**	**3.08**	**0.00**	**2.55**	**2.90**

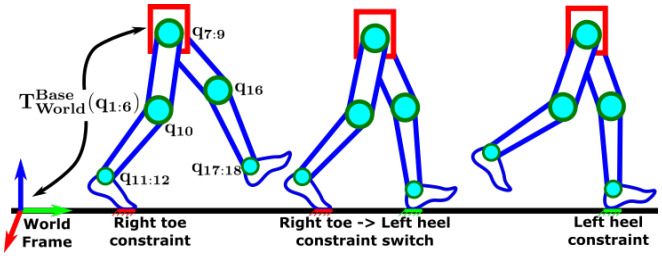


Figure 5.10 Gait estimation using alternating foot constraints. The lower body is modelled as a kinematic structure with 18 joints, 6 to allow arbitrary transformation with respect to the world frame, with an additional 3, 1, and 2 joints at the hips, knees, and ankles respectively. Left: The right toe is constrained to the ground during left leg swing phase. Middle: The constraint is switching from right toe to left heel. Right: The left heel is constrained to the ground at the beginning of right leg swing phase.

5.5.2 Switching Constraints

There are many situations where changing constraints can be imposed on human motion. For example, gait can be seen as a free floating lower body where the active constraint is alternating between heel and toe position of each foot on the ground plane (Fig. 5.10). Correctly enforcing the active constraint prevents accumulation of drift in the floating base and allows for global body position estimation. The proposed approach was tested on a 20 healthy participant dataset collected at the University of Waterloo. This experiment was reviewed by the University of Waterloo Research Ethics board, and signed consent was obtained from all participants. Data was collected using 7 IMU sensors strapped onto the thigh, calf, and foot, and torso, streaming at 100 Hz. IMU to limb rotation matrices were estimated using [167]. The motion was also simultaneously measured using a 16-camera VICON motion capture system collecting at 200 Hz to provide ground truth position and joint angle data. Each participant walked (1) in a straight line (average distance: 5.56 m ± 0.37 m), and (2) in a circle for 5 repetitions (average distance: 62.89 m ± 7.77 m). Two trials of each type of gait were captured, at the participant's own selected pace, within a bounded box in the motion capture space, for a total of 40 straight walk trajectories and 40 circle trajectories. Of these, 37 straight walk and 35 circles were used due to motion capture labelling errors.

To reconstruct walking trajectories we model gait as four alternating parent frame position constraints applied to an 18-DoF kinematic model, as described in Fig. 5.10 and visualized in Fig. 5.11. The possible position constraints are: (1) right heel, (2) left heel, (3) right toe, and (4) left toe, where the X, Y, and Z Cartesian position of the frame is the constraint. All limb link length and sensor locations were calculated from the motion capture data, while the hip rotation centre was determined via anthropometrics [165]. Similar to the biped simulation in Section 5.4,

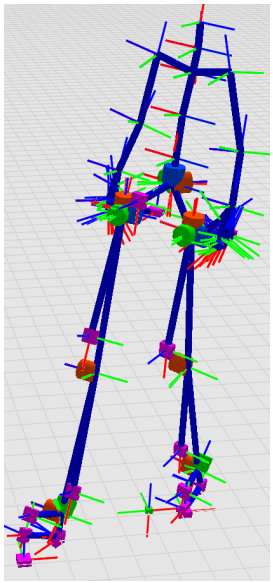


Figure 5.11 Real-data 18-DoF model, with a 6-DoF joint at the floating base, a 3-DoF joint at the hip, 1-DoF knee, and 2-DoF ankle. DoFs are denoted as coloured cylinders in this figure. The magenta squares denote the location of sensors, or a constraint point.

the inactive constraints are updated by projecting the estimated position of the feet to the ground plane at the end of each algorithm iteration. The same process and measurement noise parameters are used as in the simulation. The α parameter was tuned to be 2 for the 18-DoF model. Finally, in order to prevent fast constraint switching while standing still in double support, we impose a requirement that a constraint is active for a minimum of 20 time-steps before it can switch.

The motion capture model and the proposed method used identical kinematic models, which included a floating prismatic base. The comparison method uses the same kinematic model with a 3-DoF revolute floating base but it does not have the capability to incorporate a floating prismatic base. To time-align the IMU and the motion capture data, a squat is performed at the start and the end of each trajectory. The joint angle peaks generated by these squats are used to temporally shift and scale the two signals so they are aligned in time allowing for proper joint angle comparison. The starting floating base position, orientation, and joint angles are determined by the first frame of the motion capture data.

To analyze the accuracy of the proposed algorithm, several different error metrics are generated: (1) floating base prismatic mean Euclidean error (Eqn. 5.26, Err_{prisFb}), which provides the Cartesian error of the floating base position between the motion capture p_{moc} and the proposed algorithm p_{imu} over time t and DoFs i. This metric describes the position error

of the torso over the entire trajectory of straight and curved walking. However, it is difficult to compare the error between the two motion paths since one is straight and one is not. A second set of metrics were introduced to capture the error along the path, in the form of (2) linear distance final Euclidean error (Eqn. 5.25, Err_{LDf}). The linear distance metric calculates the linear distance travelled from the starting point by taking the norm of the velocity of the floating base in the horizontal (x and y) directions, then integrating to get position. Note that the fixed-base comparison method does not provide a meaningful prismatic or distance error value because it does not have prismatic joints and thus the kinematic model is fixed in space at the initial Cartesian position.

The (3) floating base revolute root-mean-square error (RMSE) (Eqn. 5.27, Err_{revFb}) represents the orientation error of the floating base between the motion capture q_{moc} and the proposed algorithm q_{imu}. Unlike prismatic error, both the proposed method and the comparison method contain a 3-DoF revolute floating base, allowing the entire kinematic model to rotate based on the data from the IMUs. Lastly, the (4) mean revolute joint angle RMSE (Eqn. 5.28, Err_{rev}) is considered.

$$LD_{imu} = \int_0^T \sqrt{\sum_{i=1}^{2} \dot{p}_{imu,i}^2}$$

$$LD_{moc} = \int_0^T \sqrt{\sum_{i=1}^{2} \dot{p}_{moc,i}^2}$$

$$Err_{RMSE,q} = \sqrt{\frac{1}{T}\sum_{t=0}^{T}(q_{imu,i,t} - q_{moc,i,t})^2}$$

$$Err_{LDf} = \sqrt{\sum_{i=1}^{n}(LD_{imu,T} - LD_{moc,T})^2} \tag{5.25}$$

$$Err_{prisFb} = \frac{1}{T}\sum_{t=0}^{T}\sqrt{\sum_{i=1}^{n}(p_{imu,i,t} - p_{moc,i,t})^2} \tag{5.26}$$

$$Err_{revFb} = \frac{1}{n_{fb}}\sum_{i=1}^{n_{fb}} Err_{RMSE,i} \tag{5.27}$$

$$Err_{rev} = \frac{1}{n_{rev}}\sum_{i=1}^{n_{rev}} Err_{RMSE,i} \tag{5.28}$$

Table 5.3 Comparison between the fixed-base comparison method [1] and the proposed method for straight and circle walking, showing average error in the floating base prismatic and revolute joints, as well as the body joint angles. Final position/orientation error is also given in linear distance in both magnitude and percentage error to highlight the relative error with respect to distance travelled. The comparison method does not have prismatic error reported due to its fixed-base model.

Metric			Straight Walk		Circle Walk	
			Comparison	Proposed	Comparison	Proposed
Linear Distance Err	Err_{LDf}	Final [m]	-	0.30 ± 0.39	-	2.95 ± 2.41
		Final [%]	-	5.45 ± 7.13		4.77 ± 4.11
Floating Base Pris	Err_{prisFb}	X [m]	-	0.24 ± 0.22	-	0.65 ± 0.36
		Y [m]	-	0.22 ± 0.17	-	0.60 ± 0.34
		Z [m]	-	0.02 ± 0.01	-	0.02 ± 0.01
Floating Base Rev	Err_{revFb}	R [°]	6.57 ± 4.30	5.79 ± 4.27	14.46 ± 28.61	6.81 ± 2.63
		P [°]	5.97 ± 2.78	4.88 ± 2.38	14.51 ± 28.82	6.55 ± 2.62
		Y [°]	9.70 ± 5.07	5.64 ± 5.06	86.35 ± 59.04	33.53 ± 21.83
Joint Angle Rev Left Leg	Err_{rev}	Hip R [°]	9.37 ± 5.08	7.57 ± 4.71	9.73 ± 5.34	7.66 ± 3.72
		Hip P [°]	4.57 ± 1.71	4.78 ± 1.58	6.04 ± 1.94	5.69 ± 1.41
		Hip Y [°]	6.59 ± 1.95	6.79 ± 2.68	6.87 ± 1.77	7.87 ± 5.51
		Knee R [°]	7.72 ± 2.93	5.76 ± 2.75	8.92 ± 3.85	6.70 ± 3.36
		Ankle R [°]	6.34 ± 2.60	6.30 ± 2.15	9.69 ± 4.62	9.44 ± 4.11
		Ankle Y [°]	9.71 ± 3.38	9.38 ± 3.14	11.97 ± 3.87	11.35 ± 3.97
Joint Angle Rev Right Leg	Err_{rev}	Hip R	9.68 ± 4.67	7.97 ± 4.54	9.56 ± 5.56	8.13 ± 3.92
		Hip P [°]	4.55 ± 1.68	4.58 ± 1.44	6.10 ± 2.16	5.60 ± 1.54
		Hip Y [°]	6.26 ± 2.29	5.89 ± 2.63	7.24 ± 1.96	7.48 ± 5.29
		Knee R [°]	8.78 ± 3.32	6.54 ± 3.14	10.74 ± 3.76	7.88 ± 3.44
		Ankle R [°]	7.12 ± 3.36	6.91 ± 3.29	10.77 ± 4.70	10.04 ± 4.51
		Ankle Y [°]	10.13 ± 4.60	9.56 ± 4.60	13.27 ± 8.06	12.55 ± 8.14

For metrics that accumulate over time, such as linear distance for straight and circle walking, as well as the floating base revolute yaw joint, a percentage error of the final IMU estimated position/orientation with respect to the motion capture estimated position/orientation is given, to provide some context of the accumulated error over the full trajectory.

5.5.2.1 Results

Table 5.3 shows the linear distance error, 3D prismatic and revolute error, as well as the joint angle error between the motion capture ground truth and the proposed method. A major strength of the proposed approach is the ability to estimate the floating base position with accuracy (on

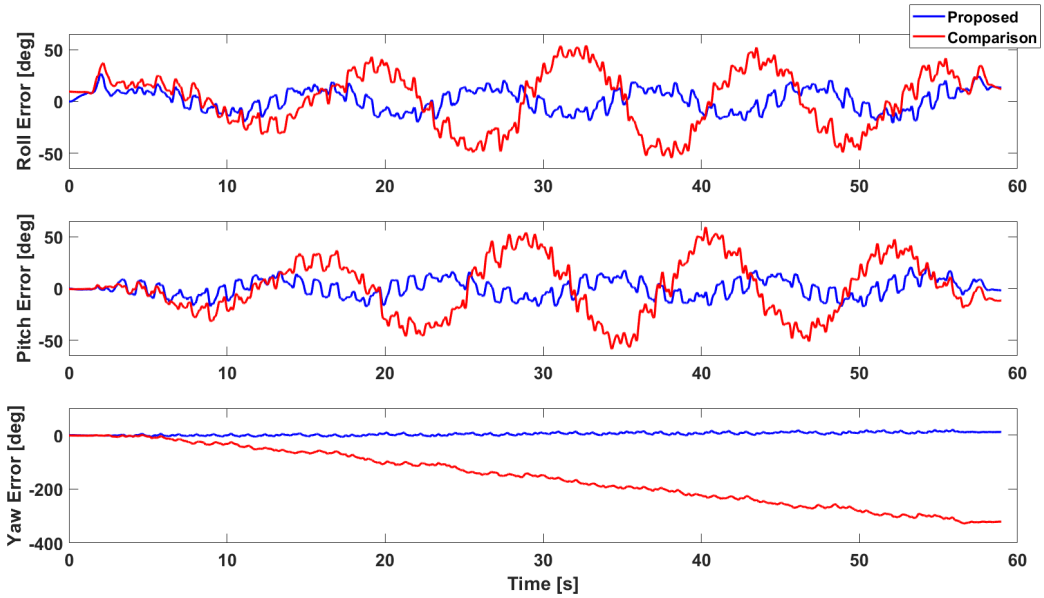


Figure 5.12 The floating base revolute joint estimate error of the proposed (blue) and the comparison (red) methods during a single example of the circle walking exercise. The significant drift in the yaw axes orientation over time is caused by integration error. For the comparison method, the accumulated error from the lack of prismatic modelling also contributes the drift. For this example, the mean roll, pitch and yaw angle error is 10, 8, and 7° for the proposed and 27, 26 and 186° for the comparison.

average within 5% with over 60 m travelled). In contrast, the comparison fixed-base model cannot provide meaningful floating base position estimates.

While yaw direction floating base error is high for both the comparison and proposed method due to to lack of gravity in the yaw direction, leading to integrational drift, the proposed method reported lower error metrics than the comparison, with final yaw orientation percentage error of 3.1% for the proposed method, compared to the 7.9% for the comparison method. This is because the proposed method can correctly attribute torso translational movements into the floating prismatic joints, while the translational movements violate the fixed-base assumption of the comparison method and must be treated as sensor noise. Table 5.3, as well as Fig. 5.12 show that the floating base revolute joint errors are significantly higher in the comparison method due to these mis-attributions. These mis-attributions can also found to provide some performance improvements in the roll and pitch directions as well, by up to 8° for the circle walking.

Since the floating base orientation determines the direction that the avatar is travelling in and thus the placement of the next step, the yaw error tends to cause the avatar to turn at a sharper

angle, leading to a smaller walking radius in circle walking (Fig. 5.14, left), or for the recovered trajectory to be rotated with respect to the ground truth (Fig. 5.14, right). While the yaw error in the proposed method is relatively small compared to the comparison method (Fig. 5.12), it is not zero and can lead to Cartesian error.

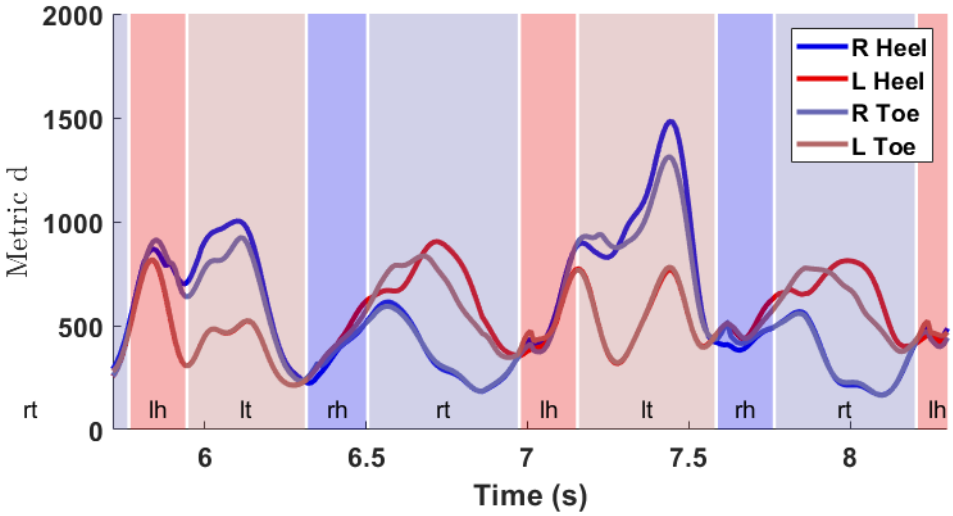


Figure 5.13 Switching constraints during a straight walk exercise. The highlights indicate the active support at that time period.

Joint angle estimates are also improved with the proposed approach, with an average error of 7.9° compared to the 8.6° in the comparison method, suggesting that while the fixed model revolute floating base absorbs a significant amount of the presumed noise from the translation, violation of the fixed translation assumption still has a negative impact on the joint angle outcomes.

When compared to the motion capture ground truth data, the proposed method tends to understep. This understepping is caused by inaccuracies in the foot switch mechanism. In simulation (Fig. 5.5), the switching is perfect due to a sharp rise in the d metric denoting when the constraint should switch to the other foot. In reality (Fig. 5.13), the switching points are much less obvious. Fig. 5.13 shows that the heel detection metric (blue, red) is often very close to the toe one (blue-purple, red-purple), especially during contract regions where the values are very similar, making it difficult to determine the perfect switching point to make the same leg heel-to-toe transition (ie Fig. 5.13, t = 6.5 s) or one leg toe to other leg heel transition (ie Fig. 5.13, t = 6.9 s). A longer look ahead horizon will allow the algorithm to determine a point where the lagging toe is clearly off the ground and the latest timestep that the foot should switch. However, even with a longer look ahead, the transition points will not be as distinct as the simulated Fig.

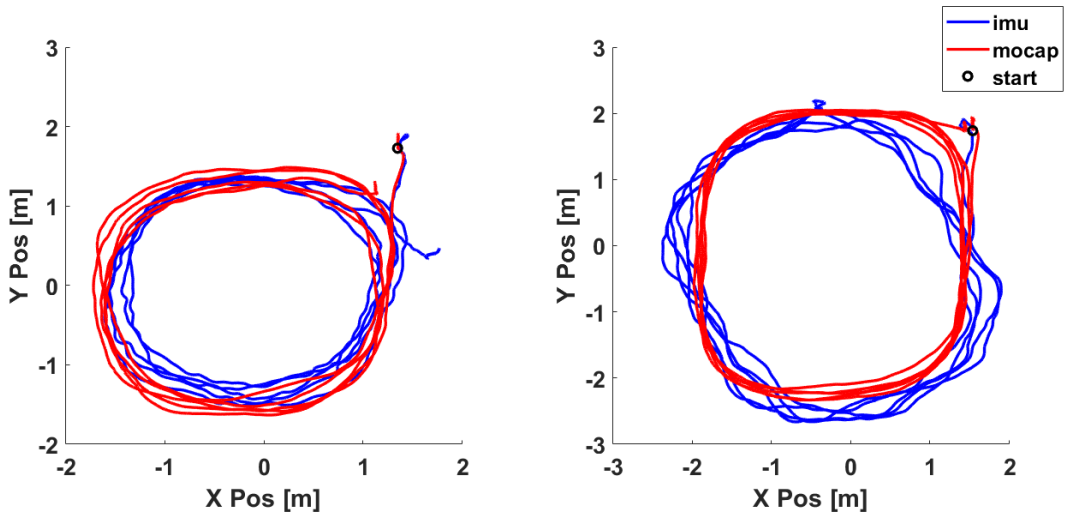


Figure 5.14 The XY position of the floating base during circular walking of two different participants. A majority of the position error between the IMU and motion capture estimates is because the IMU-based EKF tends to drift in the yaw direction when compared to ground truth data, which then increases the position error during left side of the circle, which is the opposite side of the starting position.

5.5, making it difficult to justify the higher computational cost and lag that comes with a longer look ahead.

Another potential fix is to allow multiple sets of constraints to be enabled at a time. The most simple extension of the current algorithm is to implement additional concurrent EKFs to check for the validity of right toe/heel, left toe/heel, right toe/left heel, left toe/right heel, and double support stance, effectively doubling the number of hypothesis constraints. The implementation of additional EKFs to check for combinations of constraints would significantly increase computation time.

A more time-economic route is to determine when multiple constraints are active using the existing EKFs, by relying on the d metric. This, however, is not trivial. The current algorithm selects the active constraint by choosing the EKF that generated the lowest d metric, a minimum switching time to prevent rapid switching in a small window of time, as well as a pre-determined switching order (ie left heel will always switch into left toe). This is because the d is a function of the observation residual, EKF covariance, and noise, and may be low for reasons other than when constraints are not being violated. If multiple active constraints are possible, then a threshold $d_{constraint}$ would also need to be tuned to decide when a constraint would activate, which may be different for each constraint and may change over time. These makes it difficult to determine the

active constraints using d alone.

Beyond the difficulty of knowing when to switch, another major source of error is the violation of the rigid link assumption. This is caused by sensor shifting due to skin/muscle shifting during movement, which can be significant during gait.

5.6 Summary

This chapter proposed an approach for incorporating position and orientation constraints into human pose estimation, allowing internal contact and contact with the environment to be properly modelled and exploited for improved pose estimation accuracy. Furthermore, a method to identify the most likely currently active constraint was developed which allows for accurate pose estimation even as the constraints are changing. Utilizing the changing constraints leads to IMU based global body position estimation during gait without the need for an additional modality. The approach was validated on two datasets where the constraints were constant and on human gait where there are multiple constraint switches with every step. We showed that the proposed method significantly improved pose estimation accuracy and floating base gait estimate correctly tracked the global body position with a percentage error of 5% when compared to motion capture ground truth. The proposed approach can enforce multiple *a-priori* known constraints. However, the active constraint selection will always select a single most likely active constraint. This formulation works well for estimating walking, where is is reasonable to assume that there is always a single active constraing. However, the current approach will not work for arbitrary motions that contain an unconstrained flight phase, such as running or jumping, or with motions where there might be a varying number of active constraints.

Chapter 6

Fast Approximate Multivariate Gaussian Processes (FAMGP)

A version of this chapter has been submitted to IEEE Intelligent Systems.

In this chapter a novel approximation to multivariate Gaussian processes is proposed. Gaussian processes regression models are an appealing machine learning method as they learn expressive non-linear models from exemplar data with minimal parameter tuning and estimate both the mean and covariance of unseen points. However, exponential computational complexity growth with the number of training samples has been a long standing challenge. During training, one has to compute and invert an $N \times N$ kernel matrix at every iteration. Regression requires computation of an $m \times N$ kernel where N and m are the number of training and test points respectively. The proposed approach approximates the covariance kernel using eigenvalues and functions leading to an approximate Gaussian process with significant reduction in training and regression complexity. Training with the proposed approach requires computing only a $N \times n$ eigenfunction matrix and a $n \times n$ inverse where n is a selected number of eigenvalues. Furthermore, regression now only requires an $m \times n$ matrix. Previously, Williams and Seeger proposed using the Nystrom method to estimate an eigen decomposition of a covariance kernel with known hyperparameters, showing significant speedup over conventional Gaussian processes regression [129]. Instead of approximating the eigenvalues and functions we show that for some kernels closed form solutions are available allowing us to utilize the approximation in both training and regression. In a special case the hyperparameter optimization is completely independent from the number of training samples. The proposed method can regress over multiple outputs, estimate the derivative of the regressor of any order, and learn the correlations between them. The computational complexity

reduction, regression capabilities, and multi-output correlation learning are demonstrated in simulation examples. The efficient regression of the method allows us to utilize it in real time control in a learning from demonstration framework presented in the next chapter.

6.1 Approximate Kernel Gaussian Processes

Recall that Gaussian processes introduced in Section 3.3 are a general mean function and covariance modeling method that can be trained on input output pairs. However, as discussed in Section 3.3.2 since both training and regression computational complexity are dependent on the number of training points Gaussian processes have been limited to small training datasets and non real-time regressions applications.

This section first shows how approximating the kernel matrix using n eigen functions and values leads to an approximate Gaussian process where the necessary matrix inversion is reduced from a $N \times N$ to $n \times n$. Next, it demonstrates that taking the derivative of the eigen functions also allows for estimating the k_{th} order derivative of the approximate GP. Finally, investigating the optimization of kernel hyperparameters using gradient descent it shows that computational complexity grows linearly with the number of training points as opposed to exponentially in regular GP formulation. Furthermore, in the special case when hyper parameters are present only in eigenvalues, the optimization is independent from the number of training points.

Mercer's theorem states that for any continuous symmetric non-negative definite kernel there exists an orthonormal basis consisting of eigen functions $\Phi_i(x)$ and non-increasing eigen values λ_i [168] such that

$$k(x, x') = \sum_{n=1}^{\infty} \lambda_i \phi_i(x) \phi_i(x') \tag{6.1}$$

Let us assume that we know this decomposition for our desired kernel, we can thus approximate $K_{\mathrm{x}\mathrm{x}'}$ by utilizing only n eigen values. In vector notation

$$K_{xx} \approx \Phi_{\mathrm{x}} \Lambda {\Phi_{\mathrm{x}'}}^T \tag{6.2}$$

where $\Phi_{\mathrm{x}i,j} = \phi_j(x_i) | j \in 1 \dots n$ and Λ is a diagonal matrix of the eigenvlaues $[\lambda_1, \lambda_2 \dots \lambda_n]$. Substituting this approximation into the GP prediction equations 3.34 and 3.35,

$$\mu_* \approx \Phi_{\mathrm{x}_*} \Lambda \Phi_{\mathrm{x}}^T (\Phi_{\mathrm{x}} \Lambda \Phi_{\mathrm{x}}^T + \Sigma_N)^{-1} \mathrm{y} \tag{6.3}$$

$$\Sigma_* \approx \Phi_{\mathrm{x}_*} \Lambda \Phi_{\mathrm{x}_*}^T - \Phi_{\mathrm{x}_*} \Lambda \Phi_{\mathrm{x}}^T (\Phi_{\mathrm{x}} \Lambda \Phi_{\mathrm{x}}^T + \Sigma_N)^{-1} \Phi_{\mathrm{x}} \Lambda \Phi_{\mathrm{x}_*}^T \tag{6.4}$$

Recall the binomial inverse theorem

$$(A + UBV)^{-1} = A^{-1} - A^{-1}U(B^{-1} + VA^{-1}U)^{-1}VA^{-1} \tag{6.5}$$

which allows us to simplify the inverse of $K_\Phi = (\Phi_\mathrm{x}\Lambda\Phi_\mathrm{x}^T + \Sigma_N)$ as

$$K_\Phi{}^{-1} = \Sigma_N^{-1} - \Sigma_N^{-1}\Phi_\mathrm{x}(\Lambda^{-1} + \Phi_\mathrm{x}^T\Sigma_N^{-1}\Phi_\mathrm{x})^{-1}\Phi_\mathrm{x}^T\Sigma_N^{-1} \tag{6.6}$$

Using this approximation, inference only requires the inverse of an $n \times n$ matrix $\bar{\Lambda} = \Lambda^{-1} + \Phi_\mathrm{x}^T\Sigma_N^{-1}\Phi_\mathrm{x}$. Substituting this result into the approximate prediction equations 6.3 and 6.4 leads to significantly faster prediction compared to regular GP, single output prediction equations are summarized in table 6.1.

Table 6.1 Comparison of the proposed FAMGP and regular Gaussian Process regression equations and their respective matrix sizes when predicting the mean μ_* and covariance Σ_* of the output $\mathrm{y} \in \mathbb{R}^{m\times 1}$ at m points $x_* = [x_*^1\ x_*^2\ \ldots x_*^m]^T$.

	FAMGP	GP	
Mean	$\mu_* = \underbrace{\Phi_{\mathrm{x}_*}}_{m\times n}\underbrace{\alpha'}_{n\times 1}$ $\alpha' = \Lambda\Phi_\mathrm{x}^T(\Sigma_N^{-1} - \Sigma_N^{-1}\Phi_\mathrm{x}\bar{\Lambda}^{-1}\Phi_\mathrm{x}^T\Sigma_N^{-1})\mathrm{y}$	$\mu_* = \underbrace{K_{\mathrm{x}_*\mathrm{x}}}_{m\times N}\underbrace{(K_{xx} + \Sigma_N)^{-1}\mathrm{y}}_{N\times 1}$	
Cov	$\Sigma_* = \underbrace{\Phi_{\mathrm{x}_*}}_{m\times n}\underbrace{G}_{n\times n}\underbrace{\Phi_{\mathrm{x}_*}^T}_{n\times m}$ $G = \Lambda\Phi_\mathrm{x}^T(\Sigma_N^{-1} - \Sigma_N^{-1}\Phi_\mathrm{x}\bar{\Lambda}^{-1}\Phi_\mathrm{x}^T\Sigma_N^{-1})\Phi_\mathrm{x}\Lambda$	$\Sigma_* = \underbrace{K_{\mathrm{x}_*\mathrm{x}_*}}_{m\times m} - \underbrace{K_{\mathrm{x}_*\mathrm{x}}}_{m\times N}\underbrace{(K_{xx} + \Sigma_N)^{-1}}_{N\times N}\underbrace{K_{\mathrm{x}\mathrm{x}_*}}_{N\times m}$	
Terms	Φ_{x_*} : Eigen function of prediction points x_* Φ_x : Eigen function of training points x Λ : Kernel eigen values n : Number of selected eigen values Σ_N : Training output noise covariance matrix $\bar{\Lambda} = \Lambda^{-1} + \Phi_\mathrm{x}^T\Sigma_N^{-1}\Phi_\mathrm{x}$	$K_{X_*\mathrm{x}}$: Kernel between prediction points x_* and training points x K_{XX} : Kernel between training points x y : Training outputs Σ_N : Training output covariance matrix	

6.1.1 Differentiation

Since differentiation is a linear operator, the derivative of the GP output with respect to the input is also a GP [169]. Consider two test points x_* and $x_* + \delta$, the respective outputs are then random

variables as follows:

$$y_* = \Phi_{x_*}\alpha' + \epsilon_* \tag{6.7}$$

$$y_\delta = \Phi_{x_*+\delta}\alpha' + \epsilon_\delta \tag{6.8}$$

where $\epsilon_*,\ \epsilon_\delta \sim \mathcal{N}(0, \Sigma_N^2)$. The two random variables will have a jointly Gaussian distribution

$$\begin{bmatrix} y_* \\ y_\delta \end{bmatrix} \sim \mathcal{N}\left(\begin{matrix} \Phi_{x_*}\alpha' \\ \Phi_{x_*+\delta}\alpha' \end{matrix}\middle| \begin{matrix} \Phi_{x_*}G\Phi_{x_*}^T & \Phi_{x_*}G\Phi_{x_*+\delta}^T \\ \Phi_{x_*+\delta}G\Phi_{x_*} & \Phi_{x_*+\delta}G\Phi_{x_*+\delta}^T \end{matrix}\right) \tag{6.9}$$

The derivative is thus

$$\frac{\partial y_*}{\partial x_*} = \lim_{\delta\to 0}\frac{\Phi_{x_*+\delta}\alpha' - \Phi_{x_*}\alpha'}{\delta} + \lim_{\delta\to 0}\frac{\epsilon_\delta - \epsilon_*}{\delta} \tag{6.10}$$

$$= \underbrace{\frac{\partial \Phi_*}{\partial x_*}\alpha'}_{mean} + \underbrace{\lim_{\delta\to 0}\frac{\epsilon_\delta - \epsilon_*}{\delta}}_{variance} \tag{6.11}$$

Now we substitute the variance and covariance estimate from the jointly Gaussian distribution for the sum

$$\begin{aligned} \mathbb{V}ar(\lim_{\delta\to 0}\frac{\epsilon_\delta - \epsilon_*}{\delta}) &= \lim_{\delta\to 0}\frac{1}{\delta^2}\Big(\mathbb{V}ar(\epsilon_\delta) + \mathbb{V}ar(\epsilon_*) - \mathbb{C}ov(\epsilon_\delta, \epsilon_*) - \mathbb{C}ov(\epsilon_*, \epsilon_\delta)\Big) \\ &= \lim_{\delta\to 0}\frac{1}{\delta^2}\Big(\Phi_{x_*+\delta}G\Phi_{x_*+\delta}^T + \Phi_{x_*}G\Phi_{x_*}^T - \Phi_{x_*+\delta}G\Phi_{x_*} - \Phi_{x_*}G\Phi_{x_*+\delta}^T\Big) \\ &= \frac{\partial \Phi_*}{\partial x_*}G\frac{\partial \Phi_*}{\partial x_*}^T \end{aligned} \tag{6.12}$$

Thus, if $\frac{\partial^k \Phi_{X_*}}{\partial {X_*}^k}$ is known we can compute the mean and variance for the k_{th} derivative.

$$\frac{\partial^k \mu_*}{\partial {X_*}^k} = \frac{\partial^k \Phi_{X_*}}{\partial {X_*}^k}\alpha' \tag{6.13}$$

$$\mathbb{V}ar(\frac{\partial^k y_*}{\partial {X_*}^k}) = \frac{\partial^k \Phi_{X_*}}{\partial {X_*}^k}G\frac{\partial^k \Phi_{X_*}}{\partial {X_*}^k}^T \tag{6.14}$$

In Section 6.3 we show how the structure of some available eigen functions allows for very fast computation of the derivatives.

6.1.2 Hyperparameter Training

We now consider the gradient required to optimize the hyper parameters of the approximate kernel.

$$\frac{\partial log(P(\mathrm{y}|\mathrm{x}))}{\partial \theta_j} = \frac{1}{2}\mathrm{y}^T {K_\Phi}^{-1} \frac{\partial K_\Phi}{\partial \theta_j} {K_\Phi}^{-1}\mathrm{y} - \frac{1}{2} tr({K_\Phi}^{-1}\frac{\partial K_\Phi}{\partial \theta_j}) \tag{6.15}$$

where

$$\frac{\partial K_\Phi}{\partial \theta_j} = \frac{\partial \Phi_X}{\partial \theta_j}\Lambda {\Phi_X}^T + \Phi_X \frac{\partial \Lambda}{\partial \theta_j}{\Phi_X}^T + \Phi_X \Lambda \frac{\partial {\Phi_X}^T}{\partial \theta_j} \tag{6.16}$$

Thus typical gradient descent hyperparameter optimization would require computing Φ_X, $\frac{\partial \Phi_\mathrm{x}}{\partial \theta_j}$, and the inverse of an $n \times n$ matrix at each iteration, avoiding calculating the full $N \times N$ matrix K and its inverse. Thus the computational complexity grows linearly with the number of training pairs. Any gradient descent algorithm can be utilized for parameter optimization.

Consider a special case when the hyperparameter θ_j only appears in the eigen values and not the eigen functions. Then Φ_X can be treated as a constant and $\frac{\partial \Phi_\mathrm{x}}{\partial \theta_j} = 0$. Using the fact that trace is invariant under cyclic permutations the gradient can be written entirely in terms of n sized matrices and vectors.

$$\begin{aligned}\frac{\partial log(P(\mathrm{y}|\mathrm{x}))}{\partial \theta_j} =& \frac{1}{2} {}_Y\Sigma_\Phi \Big(\frac{\partial \Lambda}{\partial \theta_j} - 2\frac{\partial \Lambda}{\partial \theta_j}\bar{\Lambda}^{-1}{}_\Phi\Sigma_\Phi \\ &+ \bar{\Lambda}^{-1}{}_\Phi\Sigma_\Phi \frac{\partial \Lambda}{\partial \theta_j}{}_\Phi\Sigma_\Phi \bar{\Lambda}^{-1}\Big){}_Y\Sigma_\Phi{}^T \\ &- tr\Big(\frac{\partial \Lambda}{\partial \theta_j}(I_n - \bar{\Lambda}^{-1})_\Phi \Sigma_\Phi\Big)\end{aligned} \tag{6.17}$$

where ${}_Y\Sigma_\Phi = \mathrm{y}^T \Sigma_N^{-1}\Phi_X$ and ${}_\Phi\Sigma_\Phi = \Phi_X^T \Sigma_N^{-1}\Phi_X$ are constant $1 \times n$ vector and $n \times n$ matrix respectively. Note that ${}_\Phi\Sigma_\Phi$ is also present in $\bar{\Lambda}$. This means that to optimize the hyper parameters that only appear in the eigen values, Φ_X^T needs only to be computed once and the iterative convergence process is independent from the number of training data points. As we show in Section 6.3, this is true for various kernel decompositions.

6.2 Multi-output Extension

A simple way to handle multi-output modelling using GPs is to assume that the outputs are independent and train a separate GP for each. However, this approach cannot capture the

correlation between different outputs present in the training data. By vectorizing the multi-output training data it is possible to capture cross output correlation [134]. Consider learning a GP representation of a function with M outputs, provided the training pairs x_i, $[y_i^1 \dots y_i^M]$, re-define the training data as $\mathrm{y} = [y_1^1\ y_2^1 \dots y_N^1\ y_1^2 \dots y_N^2 \dots y_N^M]^T$, vectorizing all of the outputs. We now consider the $NM \times NM$ covariance matrix of y

$$K_f \otimes K_{XX} + \Sigma_{NM} \tag{6.18}$$

where K_f is an $M \times M$ positive symmetric definite matrix that describes output similarities and $NM \times NM$ matrix Σ_{NM} describes the observation noise that now may include covariance between outputs, $\otimes$ denotes the Kronecker product. Note that setting K_f to the identity matrix and keeping Σ_{NM} diagonal implies independent outputs similar to training a separate GP for each. Inference can be done for multiple outputs by substitution $K_f \otimes K_{X_* \mathrm{x}}$ for $K_{X_* \mathrm{x}}$. We expand on this method by including the proposed kernel approximation in the multi-output covariance and utilizing Kronecker product properties.

We use this approach with our proposed kernel approximation and show that we can achieve significant training and regression speedup. In the case of constant observation noise we can further reduce computational complexity by relying on our approximation and the Kronecker mixed-product property. Substituting the eigenfunction and eigenvalue decomposition and relying on the mixed-product Kronecker product property we can again simplify the covariance inverse.

$$\begin{aligned} K_{I\Phi} &= K_f \otimes (\Phi_X \Lambda {\Phi_X}^T) + \Sigma_{NM} \\ {K_{I\Phi}}^{-1} &= ((I_M \otimes \Phi_X)(K_f \otimes \Lambda)(I_M \otimes {\Phi_X}^T)) + \Sigma_{NM})^{-1} \\ &= \Sigma_{NM}^{-1} - \Sigma_{NM}^{-1}(I_M \otimes \Phi_X)(K_f^{-1} \otimes \Lambda^{-1} \\ &\quad + (I_M \otimes {\Phi_X}^T)\Sigma_{NM}^{-1}(I_M \otimes \Phi_X))^{-1}(I_M \otimes {\Phi_X}^T)\Sigma_{NM}^{-1} \end{aligned} \tag{6.19}$$

The required inverse is now $nM \times nM$ instead of $NM \times NM$.

Often it is assumed that the observation noise is constant at each sample and thus can be expressed as $\Sigma_{Nm} = S_M \otimes I_N$ where S_M is an $M \times M$ positive definite matrix. In this case we can further simplify the required $nM \times nM$ matrix inverse into eigen decomposition of smaller matrices and matrix multiplication. Substituting the noise covariance $S_M \otimes I_N$ into the inverse, using Kronecker mixed-product property, and following a similar approach to [170] we see that

$$\begin{aligned} &(K_f^{-1} \otimes \Lambda^{-1} + (I_M \otimes {\Phi_X}^T)(S_M \otimes I_N)^{-1}(I_M \otimes \Phi_X))^{-1} \\ &\quad = (K_f^{-1} \otimes \Lambda^{-1} + S^{-1} \otimes {\Phi_X}^T \Phi_X)^{-1} \\ &\quad = (K_f \otimes \Lambda)(S^{-1} K_f \otimes {\Phi_X}^T \Phi_X \Lambda + I_M \otimes I_n)^{-1} \end{aligned} \tag{6.20}$$

Next we apply eigen decomposition to $S^{-1}K_f = U_a D_a U_a^{-1}$ and ${\Phi_X}^T \Phi_X \Lambda = U_b D_b U_b^{-1}$ where U denotes the matrix of eigenvectors and D is a diagonal matrix of eigenvalues. Substituting the decomposition back into 6.20, expanding the result using the mixed-product property again, and finally applying the binomial inverse theorem, the inverse simplifies to the following:

$$
\begin{aligned}
&(K_f \otimes \Lambda)(U_a D_a U_a^{-1} \otimes U_b D_b U_b^{-1} + I_M \otimes I_n)^{-1} \\
&= (K_f \otimes \Lambda)[(U_a \otimes U_b)(D_a \otimes D_b)(U_a^{-1} \otimes U_b^{-1}) + I_M \otimes I_n]^{-1} \\
&= (K_f \otimes \Lambda)[I_M \otimes I_n - (U_a \otimes U_b)(D_a \otimes D_b + I_M \otimes I_n)^{-1}(U_a^{-1} \otimes U_b^{-1})] \\
&= (K_f \otimes \Lambda) - (K_f U_a \otimes \Lambda U_b) \underbrace{(D_a \otimes D_b + I_M \otimes I_n)^{-1}}_{Diagonal} (U_a^{-1} \otimes U_b^{-1})
\end{aligned}
$$

In cases of large M and n this approach can significantly decrease the computation time since it avoids the inversion of $nM \times nM$ matrix and instead only requires eigen decomposition of $n \times n$ and $M \times M$ matrices.

6.2.1 Learning K_f

Gradient descent can be utilized to learn the matrix K_f by maximizing marginal log likelihood. To guarantee that K_f remains symmetric positive definite during convergence, it can be parametrized using Cholesky decomposition as $K_f = LL^T$ where L is a lower triangular matrix [134]. Similar to the special case when hyperparameters only appear in the eigenvalues, the gradient is written entirely in terms of $1 \times nM$ vectors and $nM \times nM$ matrices and only requres an $nM \times nM$ matrix inverse at each optimization iteration.

$$
\frac{\partial log(P(\mathrm{y}|\mathrm{x}))}{\partial L} = \frac{1}{2}\mathrm{y}^T {K_{I\Phi}}^{-1} \frac{\partial K_{I\Phi}}{\partial L} {K_{I\Phi}}^{-1}\mathrm{y} - \frac{1}{2} tr({K_{I\Phi}}^{-1} \frac{\partial K_{I\Phi}}{\partial L}) \tag{6.21}
$$

where $\frac{\partial K_{I\Phi}}{\partial L} = (I_M \otimes \Phi_X)(\frac{\partial LL^T}{\partial L} \otimes \Lambda)(I_M \otimes {\Phi_X}^T)$ and $\frac{\partial LL^T}{\partial L}$ can be calculated as $\frac{\partial LL^T}{\partial L} = (I_{(nM)^2} + T)I_{nM} \otimes L$ where T is a transformation matrix such that $Tvec(L) = vec(L^T)$ [135].

6.3 Available Kernels

In this section we present some of the kernels with well known Mercer expansions, their k_{th} order derivatives with respect to the input and gradients with respect to their hyperparameters. For a more comprehensive list the reader is referred to [171]. Note that in our formulation the scaling of any kernel is handled by the K_f matrix thus we omit the commonly included scaling factors from all of the presented kernels.

6.3.1 Squared Exponential

The squared exponential covariance function

$$k_{se}(x, x') = e^{\frac{-(x-x')^2}{2l_{se}^2}} \tag{6.22}$$

is the most commonly used kernel in GP regression. It has a single hyperparameter l_{se} which controls the kernel width and its Mercer expansion is given by [172]:

$$\lambda_{se\ i} = \sqrt{\frac{\alpha_{se}^2}{\alpha_{se}^2 + \delta_{se}^2 + \eta_{se}^2}} \left(\frac{\eta_{se}^2}{\alpha_{se}^2 + \delta_{se}^2 + \eta_{se}^2} \right)^i \tag{6.23}$$

$$\Phi_{se\ i}(x) = \sqrt{\frac{\beta_{se}}{i!}} e^{-\alpha_{se}^2 x^2} H_i(\sqrt{2}\alpha_{se}\beta_{se}x) \tag{6.24}$$

where $\eta_{se} = \frac{1}{\sqrt{2}l_{se}}$, $\beta_{se} = (1 + (\frac{2\eta_{se}}{\alpha_{se}})^2)^{\frac{1}{4}}$, and $\delta_{se}^2 = \frac{\alpha_{se}^2}{2}(\beta_{se}^2 - 1)$. The parameter α_{se} is a tuning global scaling factor and can be utilized to avoid numerical issues with computing an inverse with extremely small eigenvalues. $H_i(\cdot)$ denotes the i_{th} Hermite polynomial. The squared exponential kernel and its approximation using Mercer expansion are shown in figure 6.1.

One can interpret the expansion as a wavelet transform utilizing Hermitian wavelets. With this interpretation we see that the global scaling factor α_{se} in the eigen functions dilates or compresses the wavelet. Thus as the required range of x increases one must decrease the scaling factor for the kernel approximation to maintain accuracy. Since α_{se} is also present in the eigenvalue equations, this in turn causes a slower eigenvalue drop off. Considering the scaling factor together with the kernel width parameter l_{se} the implication is that one has to increase the number of eigenvalues for narrow kernels or when increasing the range of x. Finally, note that l_{se} is present in both the eigen values and functions and thus for the squared exponential kernel the hyperparameter optimization requires re-evaluating Φ_X at every iteration.

6.3.1.1 Squared Exponential Derivatives

The k_{th} derivative of $\Phi_{se\ i}(x)$ is calculated efficiently by noticing that $\frac{\partial^k e^{-\alpha_{se}^2 x^2}}{\partial x^k}$ can be evaluated recursively

$$\begin{aligned} \frac{\partial^k e^{-\alpha_{se}^2 x^2}}{\partial x^k} &= P_k e^{-\alpha_{se}^2 x^2} \\ P_0 = 1,\ P_1 &= -2\alpha_{se}^2 x \\ P_{k+1} &= -2\alpha_{se}(xP_k + (k-1)P_{k-1}) \end{aligned} \tag{6.25}$$

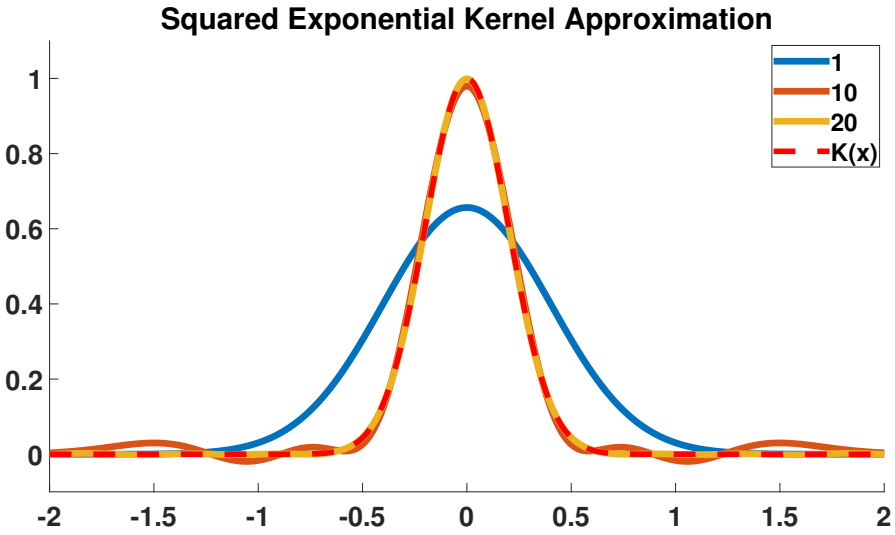


Figure 6.1 Approximation of the squared exponential kernel with $l_{se} = 0.2$ using 1, 10, and 20 eigen values. With just 20 eigenvalues the mean absolute difference between the approximation and the actual kernel values is 7.68×10^{-4}.

and $H_i(\sqrt{2}\alpha_{se}\beta_{se}x)$ represents an Appell sequence, thus

$$\frac{\partial^k H_i(\sqrt{2}\alpha_{se}\beta_{se}x)}{\partial x^k} = \frac{k!(\sqrt{2}\alpha_{se}\beta_{se})^k}{(i-k)!} H_{i-k}(\sqrt{2}\alpha_{se}\beta_{se}x) \tag{6.26}$$

Finally applying Leibniz rule we obtain the k_{th} derivative

$$\frac{\partial^k \Phi_{se\,i}(x)}{\partial x^k} = \sqrt{\frac{\beta_{se}}{i!}} \sum_{j=0}^{k} \frac{\partial^{k-j} e^{-\alpha_{se}^2 x^2}}{\partial x^{k-j}} \frac{\partial^j H_i(\sqrt{2}\alpha_{se}\beta_{se}x)}{\partial x^j} \tag{6.27}$$

6.3.1.2 Squared Exponential Hyperparameters

Kernel length l_{se} is the only hyperparameter for this kernel, the gradient of $\lambda_{se\,i}$ with respect to the kernel length is a straightforward application of the chain rule.

$$\frac{\partial \lambda_{se\,i}}{\partial l_{se}} = \left(2i\frac{\partial \eta_{se}}{\partial l_{se}} + \frac{\eta_{se}(-i-\frac{1}{2})(\frac{\partial \delta_{se}^2}{\partial l_{se}} + 2\frac{\partial \eta_{se}}{\partial l_{se}}\eta_{se})}{\alpha_{se}^2+\delta_{se}^2+\eta_{se}^2}\right)\left(\alpha_{se}\eta_{se}^{2i-1}(\alpha_{se}^2+\delta_{se}^2+\eta_{se}^2)^{-i-\frac{1}{2}}\right) \tag{6.28}$$

where

$$\frac{\partial \eta_{se}}{\partial l_{se}} = -\frac{1}{\sqrt{2} l_{se}^2} \tag{6.29}$$

$$\frac{\partial \beta_{se}}{\partial l_{se}} = \frac{2}{\alpha_{se}^2} \frac{\partial \eta_{se}}{\partial l_{se}} \eta_{se} (1 + (\frac{2\eta_{se}}{\alpha_{se}})^2)^{-\frac{3}{4}} \tag{6.30}$$

$$\frac{\partial \delta_{se}^2}{\partial l_{se}} = \alpha_{se}^2 \frac{\partial \beta_{se}}{\partial l_{se}} \beta_{se} \tag{6.31}$$

Using chain rule and relying on the Appell sequence properties of $H_i(\cdot)$, the gradient of $\Phi_{se\ i}$ with respect to the kernel length l_{se} can be evaluated efficiently as

$$\frac{\partial \Phi_{se\ i}(x)}{\partial l_{se}} = \left(\frac{1}{2\beta_{se}} \frac{\partial \beta_{se}}{\partial l_{se}} - \frac{\partial \delta_{se}^2}{\partial l_{se}} x^2 \right) \Phi_{se\ i}(x) + \sqrt{2i} \alpha_{se} \frac{\partial \beta_{se}}{\partial l_{se}} x \Phi_{se\ i-1}(x) \tag{6.32}$$

6.3.2 Periodic Kernel

The periodic kernel covariance function

$$k_{pr}(x, x') = e^{-\frac{2sin(f_{pr}\frac{(x-x')}{2})^2}{w_{pr}^2}} \tag{6.33}$$

allows to create Gaussian processes that are periodic. The frequency parameter f_{pr} determines the distance between the repetitions and the width w_{pr} controls the kernel width. The normalized Mercer expansion of the periodic kernel is derived in [173] and presents as a harmonic Fourier series.

$$\lambda_{pr\ 0} = \frac{\gamma_{pr}}{\zeta_{pr}}, \quad \Phi_{pr\ 0}(x) = 1$$

$$\lambda_{pr\ i} = \begin{cases} \frac{e^{-\frac{j^2 w^2}{2}}}{\zeta_{pr}} & j = 2i - 1 \\ \frac{e^{-\frac{j^2 w^2}{2}}}{\zeta_{pr}} & j = 2i \end{cases} \tag{6.34}$$

$$\Phi_{pr\ i}(x) = \begin{cases} cos(j f_{pr} x) & j = 2i - 1 \\ sin(j f_{pr} x) & j = 2i \end{cases} \tag{6.35}$$

where γ_{pr} and ζ_{pr} are the offset and scaling factor respectively to ensure the kernel has a range of $[0, 1]$.

$$\gamma_{pr} = \sum_{i=1}^{n} (-1)^{i-1} e^{-\frac{i^2 w_{pr}^2}{2}}, \quad \zeta_{pr} = \sum_{i=1}^{n} 2e^{-\frac{(2i-1)^2 w_{pr}^2}{2}} \tag{6.36}$$

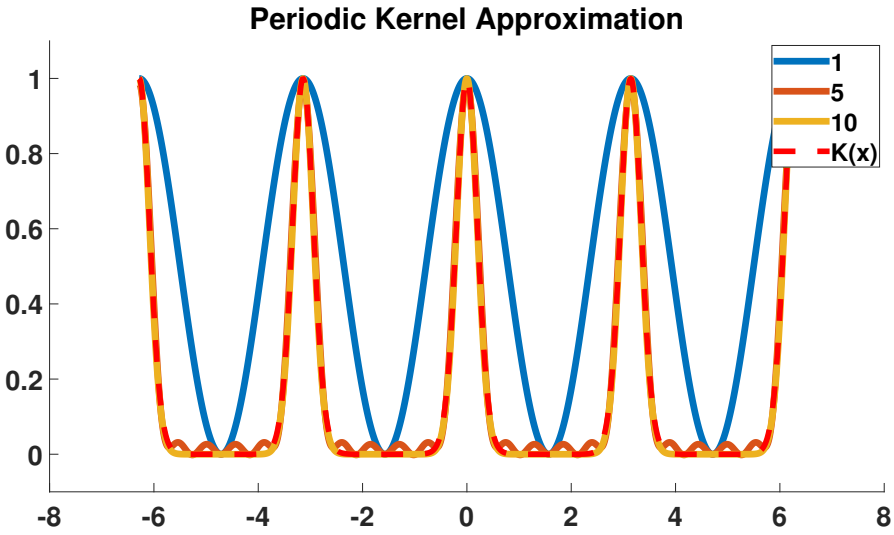


Figure 6.2 Approximation of the periodic kernel with $w_{pr} = 0.4$ and $f_{pr} = 2$ using 1, 5, and 10 eigen values. With only 10 eigenvalues the mean absolute difference between the approximation and the actual kernel values is 3.6×10^{-3}.

The periodic kernel and its approximation are shown in figure 6.2. The kernel width parameter w_{pr} only appears in the eigen values, thus when learning a GP of a signal with a known period, Φ_X does not need to be re-evaluated at every gradient descent iteration. One may also use this kernel for non-periodic signals by selecting f_{pr} such that the kernel does not repeat in the range of x.

6.3.2.1 Periodic Kernel Derivatives

The sinusoidal structure of $\Phi_{pr\ i}(x)$ leads to easy evaluation of the k_{th} derivative.

$$\frac{\partial^k \Phi_{pr\ i}(x)}{\partial x^k} = \begin{cases} -(jf_{pr})^k sin(jf_{pr}x) & j = 2i-1 \\ (jf_{pr})^k cos(jf_{pr}x) & j = 2i \end{cases} \tag{6.37}$$

Note that the above consists of scaled entries of $\Phi_{pr\ i}(x)$ and thus once Φ_X is computed, $\frac{\partial^k \Phi_X}{\partial x^k}$ can be obtained directly.

6.3.2.2 Periodic Kernel Hyperparameters

The periodic kernel frequency f_{pr} and width w_{pr} parameters only appear in the eigen functions and values respectively. Utilizing the exponential and sinusoidal structures of the eigen values and functions the necessary derivatives for gradient descent parameter optimization are as follows:

$$\frac{\partial \Phi_{pr\ i}(x)}{\partial f_{pr}} = \begin{cases} -jxsin(jf_{pr}x) & j = 2i-1 \\ jxcos(jf_{pr}x) & j = 2i \end{cases} \tag{6.38}$$

$$\frac{\partial \lambda_{pr\ 0}}{\partial w_{pr}} = \frac{1}{\zeta_{pr}}\frac{\partial \gamma_{pr}}{\partial w_{pr}} - \frac{\partial \zeta_{pr}}{\partial w_{pr}}\frac{\gamma_{pr}}{\zeta_{pr}^2} \tag{6.39}$$

$$\frac{\partial \lambda_{pr\ i}}{\partial w_{pr}} = -w_{pr}i^2\lambda_{pr\ i} - \frac{\partial \zeta_{pr}}{\partial w_{pr}}\frac{\lambda_{pr\ i}}{\zeta_{pr}^2} \tag{6.40}$$

Where $\frac{\partial \gamma_{pr}}{\partial w_{pr}}$ and $\frac{\partial \zeta_{pr}}{\partial w_{pr}}$ are derivatives of the offset and scaling factors respectively.

$$\frac{\partial \gamma_{pr}}{\partial w_{pr}} = \sum_{i=1}^{n} -(-1)^{i-1}w_{pr}i^2e^{-\frac{i^2w_{pr}^2}{2}} \tag{6.41}$$

$$\frac{\partial \zeta_{pr}}{\partial w_{pr}} = \sum_{i=1}^{n} -2w_{pr}(2i-1)^2e^{-\frac{(2i-1)^2w_{pr}^2}{2}} \tag{6.42}$$

6.3.3 Chebyshev Kernel

The last applicable kernel function we include in this work is the Chebyshev kernel [171].

$$k_{ch}(x,x') = 1 - a + \frac{2a(1-b)(b(1-b^2)-2b(x^2+x'^2)+(1+3b^2)xx')}{(1-b^2)^2+4b(b(x^2+x'^2)-(1+b^2)xx')} \tag{6.43}$$

It has two hyperparameters $a \in (0,\ 1]$ and $b \in (0,\ 1)$ and a valid Mercer expansion in the range of $x \in [-1,\ 1]$.

$$\lambda_{ch\ 0} = 1 - a,\ \lambda_{ch\ i} = \frac{a(1-b)b^i}{b} \tag{6.44}$$

$$\Phi_{ch\ 0}(x) = 1,\ \Phi_{ch\ i}(x) = \sqrt{2}T_i(x) \tag{6.45}$$

where $T_i(\cdot)$ is the i_{th} Chebyshev polynomial. The kernel and its approximation are illustrated in figure 6.3. Just like for the squared exponential kernel the expansion can be thought of as a wavelet transform, in this case using Chebyshev type wavelets. In our work this kernel function is of particular interest since all of the hyperparameters appear only in the eigen values.

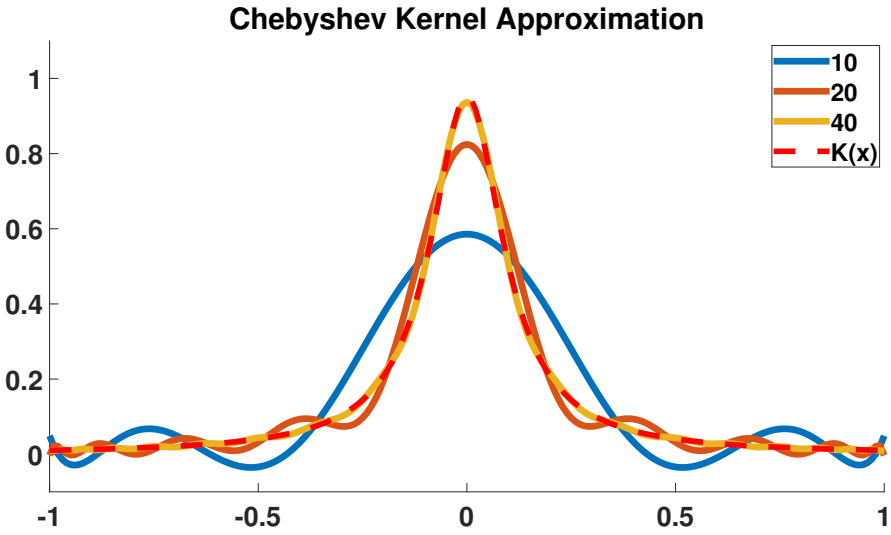


Figure 6.3 Approximation of the Chebyshev kernel with $a = 0.9$ and $b = 0.9$ using 10, 20, and 40 eigen values.

6.3.3.1 Chebyshev Kernel Derivatives

Similar to the Hermite polynomial derivatives presented in Section 6.3.1.1, the k_{th} derivative of Chebyshev polynomial can be represented through Chebyshev polynomials of lower degrees [174].

$$\frac{\partial^k T_i(x)}{\partial x^k} = 2^k \sum_{j=0}^{(i-k)/2}{}' i(i-1-j)^{\underline{k-1}} \binom{k+j-1}{k-1} T_{i-k-2j}(x) - even(i-k)2^{k-1} n(\frac{i+k}{2}-1)^{\underline{k-1}} \binom{\frac{i+k}{2}-1}{k-1} \tag{6.46}$$

Where underlined superscript indicates falling factorials $x^{\underline{n}} = x(x-1)\dots(x-n+1)$ and the $even(\cdot)$ function outputs 1 for even arguments and 0 otherwise. Since $\Phi_{ch\ i}(x)$ is obtained by scaling $T_i(x)$, $\frac{\partial^k \Phi_X}{\partial x^k}$ can be computed efficiently from Φ_X when using the Chebyshev kernel.

6.3.3.2 Chebyshev Kernel Hyperparameters

For this kernel the hyper parameters appear only in the eigen values allowing for extremely fast gradient descent based parameter optimization.

$$\frac{\partial \lambda_{ch\ 0}}{\partial a} = -1, \quad \frac{\partial \lambda_{ch\ i}}{\partial a} = \frac{\lambda_{ch\ i}}{a} \tag{6.47}$$

$$\frac{\partial \lambda_{ch\ 0}}{\partial b} = 0, \quad \frac{\partial \lambda_{ch\ i}}{\partial b} = -a(i(b-1)+1)b^{i-2} \tag{6.48}$$

6.3.4 Matern Kernel

For completeness we include also the popular Matern kernel.

$$k_m(x, x') = \frac{2^{1-v_m}}{\Gamma(v_m)}\left(\frac{\sqrt{2v_m}|x-x'|}{l_m}\right)^v K_v\left(\frac{\sqrt{2v}|x-x'|}{l_m}\right) \tag{6.49}$$

Where v_m and l_m are the hyperparameters, $K_v(\cdot)$ and $\Gamma(\cdot)$ are the modified Bessel factorial functions. The Matern kernel has an eigenvalue and function decomposition but it can be shown that the eigenvalues decay much slower than those of the squared exponential. Furthermore, the Matern kernel is only $v-1$ times differentiable and thus does not allow us to compute output derivatives of arbitrary order [175].

It is important to note that for all of the presented expansions, as the width of the kernel decreases the number of eigenvalues necessary for an accurate approximation increases. Thus our method is particularly well suited when the number of data points is significantly larger than the number of eigen functions needed to accurately approximate the kernel. Using this approach with an inadequate number of eigen functions will lead to convergence to a wider kernel than optimal.

6.4 Experiments

In this section we evaluate the computational complexity and accuracy of the proposed method. First we show that the training time of the proposed approach scales linearly when Φ_x has to be re-evaluated every training iteration and is independent from the number of training points when the hyperparameters only appear in the eigen values. Next we evaluate the accuracy of the fast approximate multi-output GP considering both the numbers of training samples and eigenvalues. Finally, we show that the scaling matrix K_f can correctly identify the correlation between outputs.

6.4.1 Computational Complexity

As discussed in Section 6.1, the proposed method requires only an inverse of $nM \times nM$ matrix instead of $NM \times NM$, where n, M, and N are the number of eigen values, outputs, and training samples respectively. During hyperparameter optimization the proposed approach further splits into two categories: (1) when the parameters are present in both eigen values and functions or (2) only in the eigen values. In the first case, Φ_x needs to be re-evaluated after every training iteration, while in the second it is treated as constant and only the eigenvalues are updated. Figure 6.4 shows the time it takes to complete 100 iterations of hyperparameter optimization using gradient descent for regular GP and the two cases of the proposed approach. As expected, regular GP quickly becomes intractable as the number of samples grows. In the proposed method, when Φ_x needs to be re-evaluated at every iteration, the computational complexity grows linearly with the number of samples in the training set. When parameters are only present in the eigenvalues, the hypoerparameter learning time is independent from the number of samples in the training dataset.

6.4.2 Accuracy

To validate the regression accuracy we generate training data from an arbitrary generating function, using a sum of sinusoids of random frequencies, amplitudes, and phase shifts. This allows us to obtain the true k_{th} derivative of the signal and verify that the proposed approach can correctly estimate high order derivatives. Zero mean Gaussian (ZMG) noise is added to the training data to simulate sensor noise. The training data is generated from:

$$\mathrm{y}_{true} = \sum_{1}^{10} c_i sin(f_i x + \varphi_i) + \epsilon_s \tag{6.50}$$

Where the amplitude coefficients c_i, frequencies f_i, and phase shifts φ_i are drawn from a uniform distribution $U(1,\ 10)$ and $\epsilon_s \sim \mathcal{N}(0, 5)$. The input variable x consists of 10000 samples evenly spaced on the interval $[-5,\ 5]$. Figure 6.5 shows the regression capabilities of the Chebyshev kernel.

As the number of training samples increases so should the regression accuracy. FAMGP allows us to utilize significantly larger training datasets. Figure 6.6 shows the RMSE with respect to the number of training samples for regular GP, and FAMGP with squared exponential and Chebyshev kernels for the data presented in figure 6.5. Due to computational complexity we are not able to utilize more than 2000 samples for the regular GP, FAMGP can easily be trained with a million,

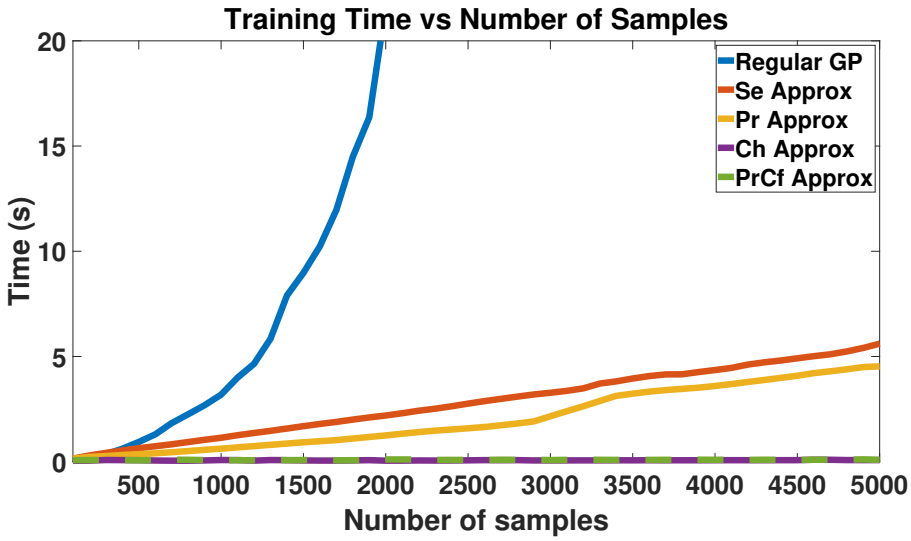


Figure 6.4 Required time to complete 100 iterations of gradient descent during hyperparameter optimization. Regular GP uses MATLAB's $fitgp$ function and we can observe cubic computational complexity with respect to the number of samples. When the proposed approach utilizes the Squared exponential (Se Approx) or Periodic (Pr Approx) kernel approximation it requires re-evaluating Φ_x at every iteration and thus the training time is directly proportional to the number of samples. Employing the Chebyshev kernel (Ch Approx) or Periodic kernel with constant frequency (PrCf Approx) approximations requires only a single evaluation of Φ_x, during training the approach only updates the eigen values. For this demonstration the input x is evenly spaced samples from (-1, 1) and the output is $sinc(x)$, 20 eigen values were used for all kernels.

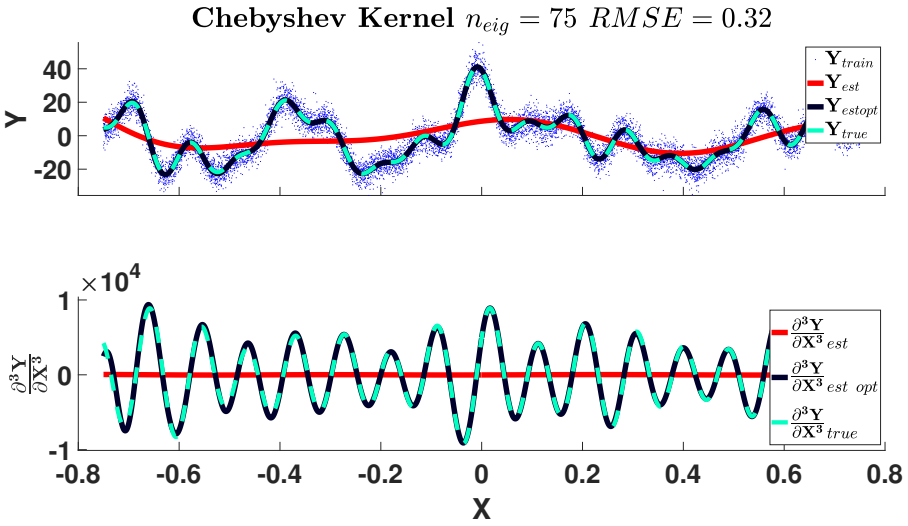


Figure 6.5 Regression of finite Fourier series with ZMG noise using FAMGP with the Chebyshev kernel approximation. In the top plot, blue dots and teal dashed line show the noisy training samples Y_{train} and the noise free signal Y_{true}, red solid line Y_{est} is the initial regression result before hyper parameter optimization ($a = 0.5,\ b = 0.5$), black line $Y_{est\ opt}$ shows the regression after optimizing the hyper parameters ($a = 0.998,\ b = 0.954$) using 5000 iterations of gradient descent which took 5.3 seconds to complete. The bottom plot shows the ability of the proposed approach to estimate the derivatives of the output, here we show estimated and actual jerk of the signal ($k = 3$).

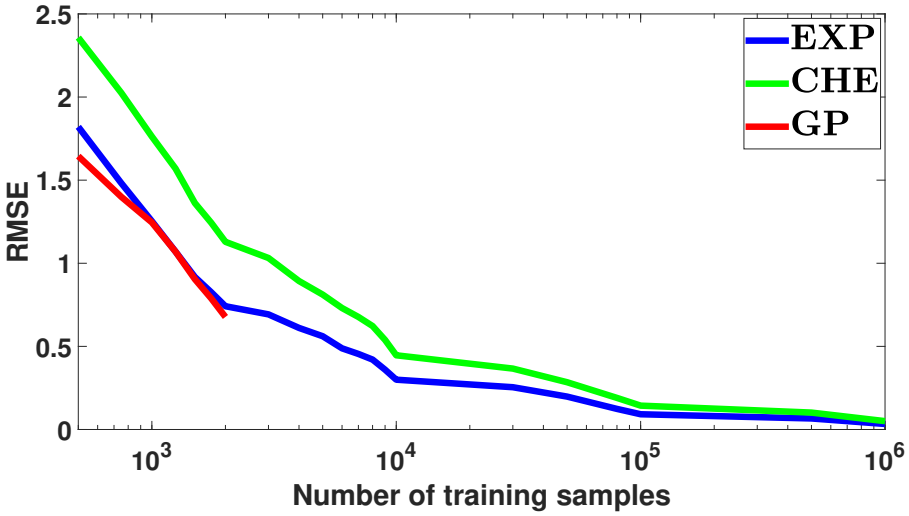


Figure 6.6 Regression accuracy improvement as the number of training samples increases. Regular GP training is not feasible for more than 2000 samples. FAMGP allows to optimize hyperparameters even with a million data points. The accuracy of both the squared exponential (EXP) and Chebyshev (CHE) kernel approximations converges as the number of samples increases. 75 eigenvalues were used for both kernels.

significantly improving the accuracy. The squared exponential kernel approximation provides lower RMSE compared to Chebyshev. However, Chebyshev kernel parameter optimization is significantly faster since Φ_x is computed only once.

Next we look at how the chosen number of eigenvalues effects the regression accuracy. We compare the performance of FAMGP with different number of eigenvalues to the standard GP formulation using the squared exponential kernel. Since the approximation can be interpreted as a wavelet transform, increasing the number of eigenvalues allows to accurately approximate a narrower kernel. Consider a sum of 10 sinusoids on the interval $x \in (-1,\ 1)$ with frequencies evenly distributed from 1 to $10 rad/s$ and ZMG noise added of standard deviation of 0.1. Figure 6.7 shows the regression RMSE as we increase the number of eigenvalues. The accuracy and kernel parameters of the proposed approach converge to that of regular GP as the number of eigenvalues increases sufficiently to correctly approximate the narrow kernel. While the squared exponential kernel is the most commonly used covariance function when using GP regression, for FAMGP, the Chebyshev kernel is particularly attractive since Φ_x does not need to be recalculated during hyperparameter optimization and allows for very fast training. The analysis shows that, while requiring more eigenvalues, the regression accuracy when using the Chebyshev kernel is comparable to that of squared exponential. Recall that regression for FAMGP requires

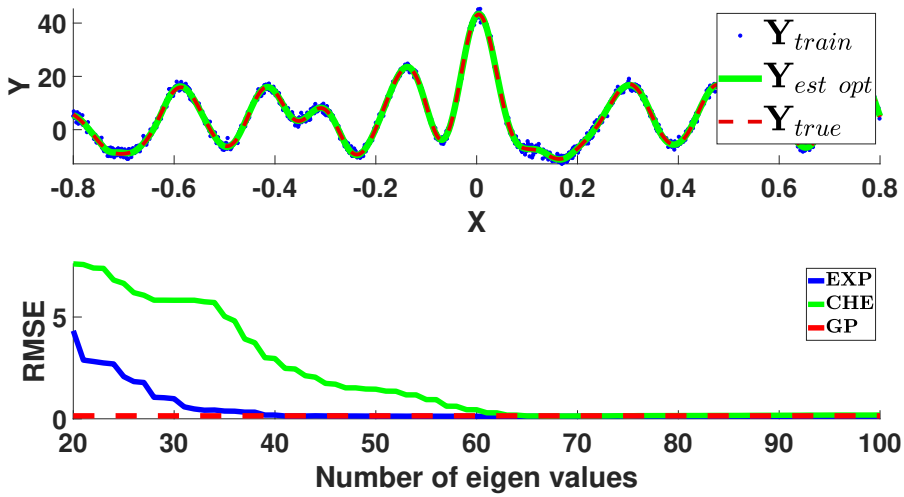


Figure 6.7 Regression RMSE as the number of eigenvalues increases. The top plot shows the noisy training data y_{train}, ground truth y_{true}, and FAMPG prediction $y_{est\ opt}$ signals when using the squared exponential kernel approximation with 100 eigenvalues. The bottom plot shows the regression RMSE as the number of eigenvalues increases from 20 to 100. The regression error of FAMGP converges to that of regular GP using both the squared exponential (EXP) and Chebyshev (CHE) kernel approximations. For the squared exponential kernel regular GP regression converges on width and scaling factors of 0.050 and 225.55 respectively, at 50 eigenvalues FAMGP optimization converged to very similar hyperparameter values $l_{se} = 0.048$ and $K_f = 215.22$, the eigenvalues sum to capture 97% of data

multiplication of the eigen function with $n \times 1$ vector and $n \times n$ matrix for mean and covariance prediction respectively. Basic matrix multiplication has cubic computational complexity with respect to the number of elements. Thus, regression for a single input is an $\mathcal{O}(n^3)$ operation for any of the kernel approximations.

6.4.3 Correlation

Finally we demonstrate that FAMGP can correctly estimate the correlation between outputs and significantly improve regression when partial outputs are available. Furthermore, we compare the multi-output performance to that of regular GP [134] and show that both methods perform equally well and converge to almost identical correlation matrix and kernel parameters. We sample 2000 training points of a highly correlated 2 dimensional signal from a zero mean normal distribution with a known covariance matrix generated utilizing equation 6.18. The squared exponential covariance (eq. 6.22) with kernel parameters $l_{se} = 0.1$ is used for K_{XX} and $x \in (-1,\ 1)$. High

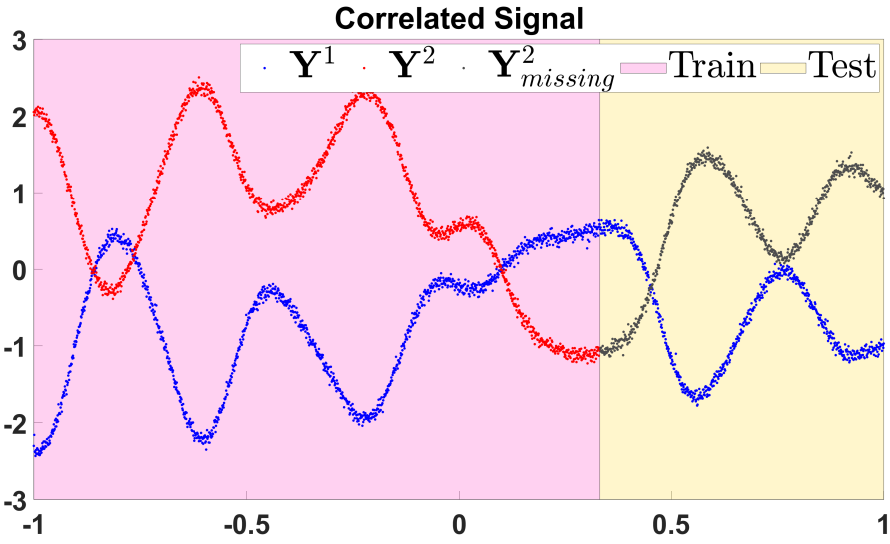


Figure 6.8 Two dimensional correlated training signal. The pink and yellow shaded regions are the training and test data sets respectively. Noisy output 1 (Y^1 blue) is available both during training and testing. Output 2 is available for training (Y^2 red) but is missing from the test set ($Y^2_{missing}$ gray).

correlation between the outputs is achieved by setting K_f as follows:

$$K_f = \begin{bmatrix} 1.0 & -0.95 \\ -0.95 & 1.0 \end{bmatrix}$$

Zero mean Gaussian noise is added to the output with $\Sigma_{Nm} = 0.05 I_{Nm}$. To test the ability of the proposed approach to utilize output correlation for regression we learn the kernel parameters and K_f using the first 1333 data points. Next, α' and G (table 6.1) are computed utilizing all 2000 samples of output 1 and only the 1333 training samples of output 2. This simulates the situation where historical data of both correlated outputs is available for training. However, during regression, we have one output and would like to estimate the other. The data is visualized in figure 6.8.

We train regular GP with the full squared exponential kernel and FAMGP with the kernel approximation utilizing 75 eigenvalues, initial kernel parameters of $l_{se} = 0.5$ and initial correlation matrix set to identity, $K_f^{init} = I_2$. Gradient descent converges on parameters shown in table 6.2. The method correctly estimates a strong negative correlation between the outputs even in the presence of significant noise. Figure 6.9 shows the FAMGP regression results over the test region when assuming independent outputs ($K_f = I_M$) and using the learned correlation matrix, clearly demonstrating the benefits of the multivariate GP extension. Table 6.3 compares the regression

accuracy of FAMGP and regular GP for training and test data regions. Using 75 eigenvalues and functions to estimate a squared exponential kernel of length 0.1 is accurate to 99.99% and thus the results between FAMGP and regular GP are almost identical. However, the training, regression, and storage requirements of FAMGP are magnitudes less than that of regular GP. For this example, at each training iteration FAMGP computes the 1333×75 Φ_X matrix and evaluates a 150×150 inverse, regular GP calculates the full 1333×1333 kernel and the inverse of a 2666×2666 matrix. The proposed approach and regular GP took 45 and 441 seconds respectively to complete the required 926 gradient descent iterations for parameter convergence. After training, FAMGP needs to only save the 150 element α' vector and 150×150 G matrix while GP needs the full 2666×2666 kernel inverse. Finally, for mean regression over the test set, FAMGP computes a 667×75 Φ_X and multiplies it with the first 75 rows of α' to estimate y^1 and last 75 rows for y^2, GP requires 667×1333 kernel calculation and multiplication of the Kronecker product of the kernel and the correlation matrix with a 2666 sized vector. The computational requirements grow linearly for FAMGP and exponentially for GP, thus while we can significantly increase the dataset size for the proposed approach, regular GP quickly becomes intractable.

Table 6.2 Optimized K_f matrix and kernel width for correlated outputs. Gradient descent converges to the true kernel width and accurately finds the negative correlation between outputs 1 and 2. The optimized parameters are very similar for both FAMGP and regular GP.

		FAMGP		GP	
		Y^1	Y^2	Y^1	Y^2
$K_f^{opt} =$	Y^1	1.567	-1.582	1.559	-1.554
	Y^2	-1.582	1.706	-1.554	1.664
$l_{se}^{opt} =$		0.108		0.109	

Table 6.3 Regression root mean squared error for the correlated data split into training, test, and entire dataset.

	Train	Test	All
FAMGP	1.28E-04	0.0421	0.0141
GP	1.54E-04	0.0418	0.0141

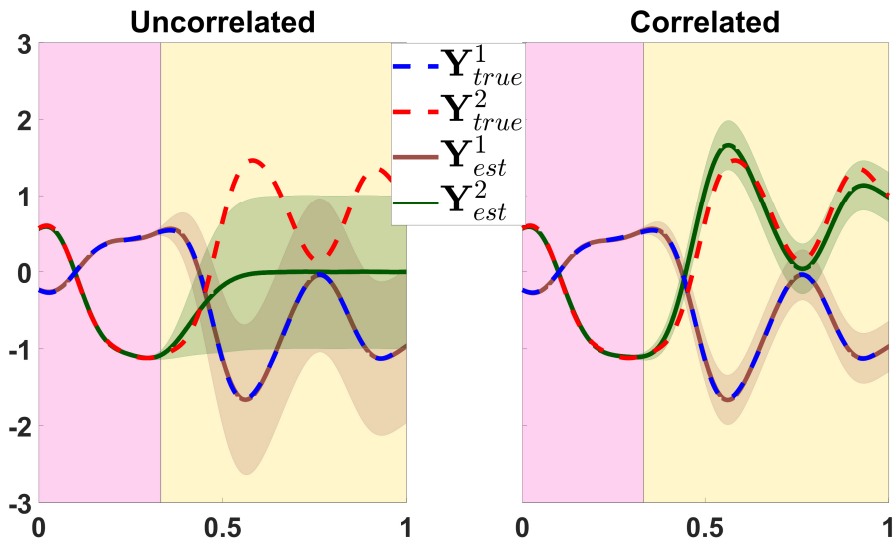


Figure 6.9 FAMGP regression over strongly correlated outputs. In the test region (yellow) noisy samples of Y^1 are available while Y^2 is entriely missing as explained in figure 6.8. Left: Uncorrelated output assumption, $K_f = \mathbf{I}_2$. When the outputs are assumed uncorrelated even though Y^1 is available for regression in the test region it is not utilized in estimation of Y^2 and the estimate drops to the zero mean assumption. Right: Using K_f learned from the training region. Due to the correlation between outputs FAMGP can utilize the Y^1 samples in estimating Y^2 and maintain regression accuracy.

6.5 Summary

In this chapter a novel fast approximate multivariate Gaussian process framework is presented. The key idea of the method is to approximate the covariance kernel using a finite number of eigenvalues and eigenfunctions. For a single output model this allows to reduce the required computational complexity of a GP training iteration from $\mathcal{O}(N^3)$ to $\mathcal{O}(n^3)$ where N and n are the number of training samples and eigenvalues respectively. In the multi-output case complexity is reduced from $\mathcal{O}((MN)^3)$ to $\mathcal{O}((Mn)^3)$ where M is the number of outputs. The proposed approach not only allows for fast training and estimation but also provides any order analytic derivatives of the GP. We provide the eigenvalues and functions of three different kernels (squared exponential, periodic, and Chebyshev) and show that in special cases hyperparameter optimization can be completely independent from the number of training samples. The method is extensively validated in simulation showing that depending on the optimal kernel width the proposed method's accuracy converges to that of regular GP with only a few eigenvalues.

Chapter 7

Controller learning from estimated motion

In this chapter a new approach to learning from demonstration using fast multi-output approximate Gaussian processes is presented. The mean and covariance predicted by the FAMGP is utilized as a time varying quadratic cost function learned from human demonstrations in a model predictive control framework. The computational efficiency of FAMGP allows us to predict the cost over a look ahead horizon, figure out the optimal control, and apply the torques to a 7DOF manipulator at 1KHz. Section 7.1 describes how by introducing a novel phase variable we re-parametirize the learned process allowing it to be used for trajectory alignment during training as well as for safe and compliant control independent of time. The FAMGP predicted mean and covariance can be used as a quadratic cost function either in terms of time of phase as described in Section 7.2. Simulations show that FAMGP is an improvement over GMM or ProMP stochastic trajectory modeling (Sec 7.3.1) and extensive validation on a real manipulator demonstrates the capabilities of the proposed learning and control method (Sec 7.3.2). Fig. 7.1 provides an overview of the proposed approach.

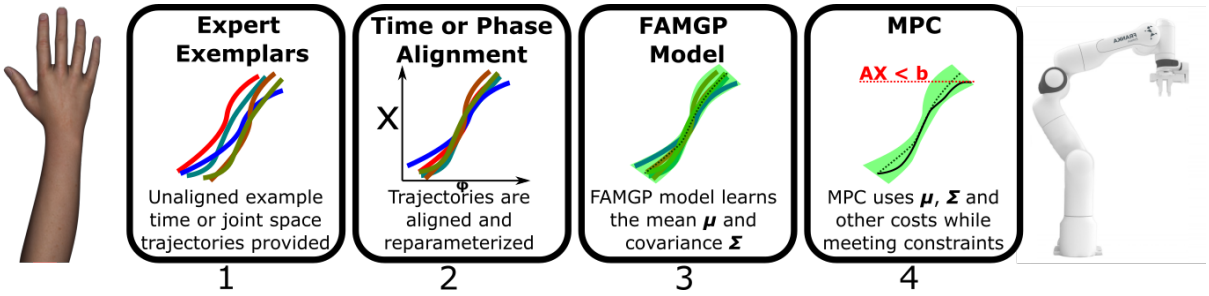


Figure 7.1 Overview of the proposed approach. 1) Multiple exemplar trajectories are provided by a human demonstrator. To get their distribution the trajectories must be aligned in time. 2) FAMGP nearest phase alignment (Sec. 7.1) is used to re-parameterize all of the trajectories with respect to consistent time $t \in [0,\ t_f\,]$ or phase $\Gamma \in [0,\ 1]$. 3) The FAMGP model is trained using the aligned exemplars and can accurately model the mean and covariance using only a few eigen vectors and values (Sec. 7.2). 4) Model predictive control (MPC) utilizes the mean and covariance as well as their derivatives as a cost function of the state (Sec. 7.1.1). The resulting controller focuses on minimizing control input and providing accurate tracking in the high and low variability regions respectively while satisfying constraints.

7.1 Training Exemplar Alignment

Before we begin learning and using a Gaussian process trajectory representation of expert motion it is of paramount importance that that the exemplar trajectories are temporally aligned. Consider N human motion exemplar trajectories $[\mathrm{y}_i, t_i]$, where y_i and t_i are the states and sample times for the ith exemplar. Due to differences in movement speeds, data collection time starts, or inconsistent sampling times ($t_i \neq t_j$), the collected exemplars may not be temporally aligned (Fig. 7.2 left). Using temporally unaligned data to learn human motion can lead inaccurate estimates of the motion variance.

We propose an iterative method to re-parameterize all of the training trajectories in terms of a single phase variable $\Gamma \in [0, 1]$ and align all of the exemplars to the phase variable. We take the first exemplar trajectory and generate the phase variable by normalizing the temporal vector.

$$\Gamma_1 = \frac{t_1 - t_1^0}{t_1^f} \tag{7.1}$$

where t_1^0 and t_1^f are the beginning and final sampling times for the first exemplar. Next, we train a multivariate Gaussian process $GP(\Gamma)$ using the pairs $\{\mathrm{y}_1, \Gamma_1\}$ and identity as the covariance matrix. The GP mean can be viewed as a smooth differentiable function that closely tracks the first exemplar trajectory y_1 in terms of our new phase variable Γ. In order to phase align all of

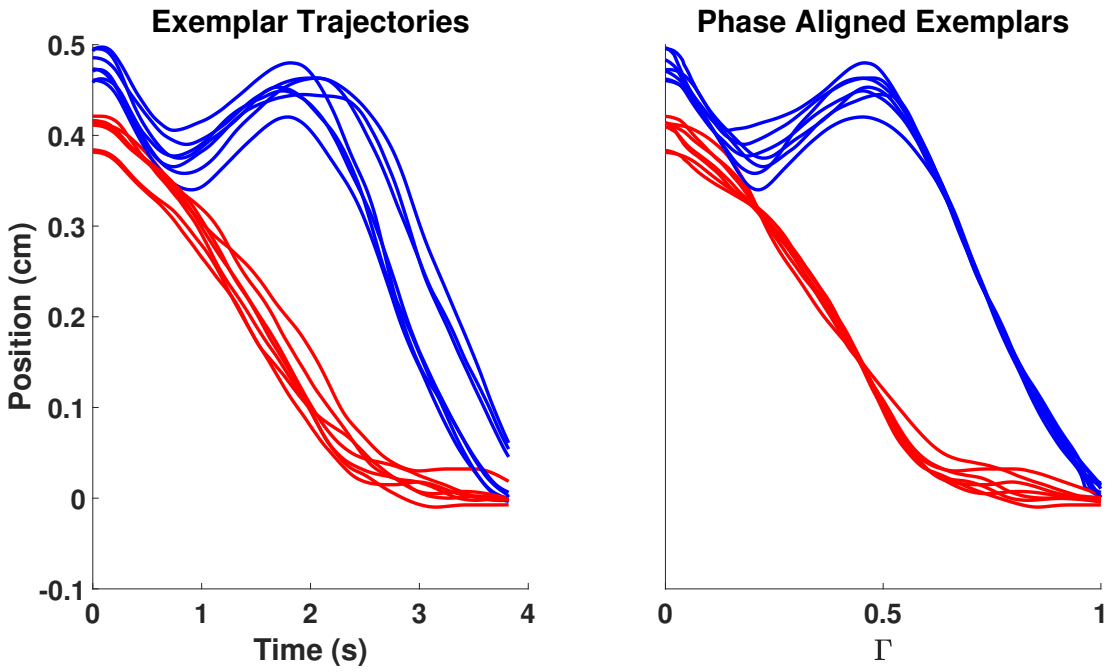


Figure 7.2 Phase parametrization and alignment of character writing trajectories. Left: original recorded time-based x (red) and y (blue) writing samples of the letter *r*. Right: proposed method phase re-parameterized aligned exemplars.

the other exemplars, for the kth state sample y_j^k in the jth exemplar we find the nearest phase Γ^* such that the error between the sample and $GP(\Gamma^*)$ is minimized.

Consider a window around the kth state sample of size H, $\mathrm{y}_j^k = [\mathrm{y}_j^{k-H/2}, \mathrm{y}_j^{k+1-H/2}, \ldots, \mathrm{y}_j^{k+H/2}]^T$. We formulate a minimization problem to find Γ^* such that the difference between the sample window y_j^k and $GP(\Gamma^*)$ is minimized.

$$\begin{aligned} e &= vec(GP(\Gamma) - \mathrm{y}_j^k) \\ \begin{bmatrix} \Gamma^* \\ s^* \end{bmatrix} &= \arg\min_{\Gamma, s} \frac{1}{2} e^T W e \end{aligned} \tag{7.2}$$

Where $\Gamma = [\Gamma - s\frac{H}{2} \ldots \Gamma + s\frac{H}{2}]$ is a phase window of size H centered at Γ and sampled at intervals of s and $vec(\cdot)$ stacks an $H \times M$ matrix into a $HM \times 1$ vector. W is a positive definite weighing matrix. Including the phase window sampling interval in the minimization allows us to align exemplars with different movement speeds.

Note that analytical derivatives of $GP(\Gamma)$ with respect to Γ are available as described in Section 6.1.1. Applying chain rule first to (eq. 7.2) and next to $GP(\Gamma)$ provides the first and second derivative of eq. 7.2 with respect to Γ.

$$\frac{\partial \frac{1}{2} e^T W e}{\partial \Gamma} = e^T W \frac{\partial GP(\Gamma)}{\partial \Gamma} \tag{7.3}$$

$$\frac{\partial^2 \frac{1}{2} e^T W e}{\partial \Gamma^2} = \frac{\partial GP(\Gamma)}{\partial \Gamma} W \frac{\partial GP(\Gamma)}{\partial \Gamma}^T + e^T W \frac{\partial^2 GP(\Gamma)}{\partial \Gamma^2} \tag{7.4}$$

Similarly the derivatives with respect to interval s are computed:

$$\frac{\partial GP(\Gamma)}{\partial s} = \frac{\partial GP(\Gamma)}{\partial \Gamma} \odot \begin{bmatrix} -H/2 & \ldots & H/2 \end{bmatrix}^T \tag{7.5}$$

$$\frac{\partial^2 GP(\Gamma)}{\partial s^2} = \frac{\partial^2 GP(\Gamma)}{\partial \Gamma^2} \odot \begin{bmatrix} (-H/2)^2 & \ldots & (H/2)^2 \end{bmatrix}^T \tag{7.6}$$

where $\odot$ indicates element wise multiplication. Newton-Raphson method is used to iteratively converge to Γ^* and $s*$ for each sample window y_j^K in the jth exemplar.

$$\begin{aligned} \Delta\Gamma = \left. \frac{\frac{\partial e^T W e}{\partial \Gamma}}{\frac{\partial^2 e^T W e}{\partial \Gamma^2}} \right|_{\Gamma_k^*, s_k^*}, \quad \Delta s = \left. \frac{\frac{\partial e^T W e}{\partial s}}{\frac{\partial^2 e^T W e}{\partial s^2}} \right|_{\Gamma_k^*, s_k^*} \\ \begin{bmatrix} \Gamma^* \\ s^* \end{bmatrix}_{k+1} = \begin{bmatrix} \Gamma^* \\ s^* \end{bmatrix}_k - \begin{bmatrix} \Delta\Gamma \\ \Delta s \end{bmatrix} \end{aligned} \tag{7.7}$$

Applying the method to all of the trajectories ensures they are aligned to the first exemplar and parameterized by $\Gamma_j \in [0,\ 1]$. Finally, cubic spline interpolation is applied to each phase parameterized exemplar to re-sample the data such that the phase vectors of all trajectories are matching. This leads to easy computation of the mean and covariance of the training data at each phase sample Γ^k. Fig. 7.2 shows the method aligning 2D character writing trajectories [176] for the letter r. It is clear that the original trajectories are not temporally aligned, the peaks of both x and y coordinates do not occur at a consistent time. The proposed approach accurately aligns the exemplars and allows for better mean and variance estimates. Fig. 7.3 shows the mean and covariance estimated by the GP after training using the mean and covariance at each aligned data sample. The need for including the phase sampling interval s in the minimization is shown in Fig. 7.4. When aligning trajectories which are similar but may have been collected at different velocities, adjusting the sampling interval allows to stretch or squeeze the error window leading to better Γ^* estimates. Without this flexible window the algorithm would have a hard time converging to a low error in windows near the peaks of the sinusoids as they would not closely overlap. Note that the approach can be used to re-parameterize from time to phase or for temporal alignment.

7.1.1 Application to Control

The above approach also allows us to re-parameterize a control task from time to the phase variable. In real world applications, time based control may not always be desirable since the control signal continues to increase if the system encounters an obstacle. For example, consider a collaborative task between a human and a manipulator, if the robot encounters human intervention during the desired motion we would like the controller to hold the manipulator in place until the intervention is removed and then continue through the learned motion from the current state. Given the current state of an m dimensional dynamic system $x' = [x_1\ x_2 \cdots x_m]^T$ and a previously learned Gaussian process $x(\Gamma) \sim GP(\Gamma)$, we can use the Newton-Raphson method to find Γ^* which minimizes the distance between x' and $x^* = GP(\Gamma^*)$. The desired control signal u^* can then be calculated and applied based on x', x^*, and Γ^*. If the interference is still present and the manipulator is not able to move, at the next control loop iteration the nearest phase would remain Γ^*, u^* would be applied again, preventing the control signal from continuously growing with time.

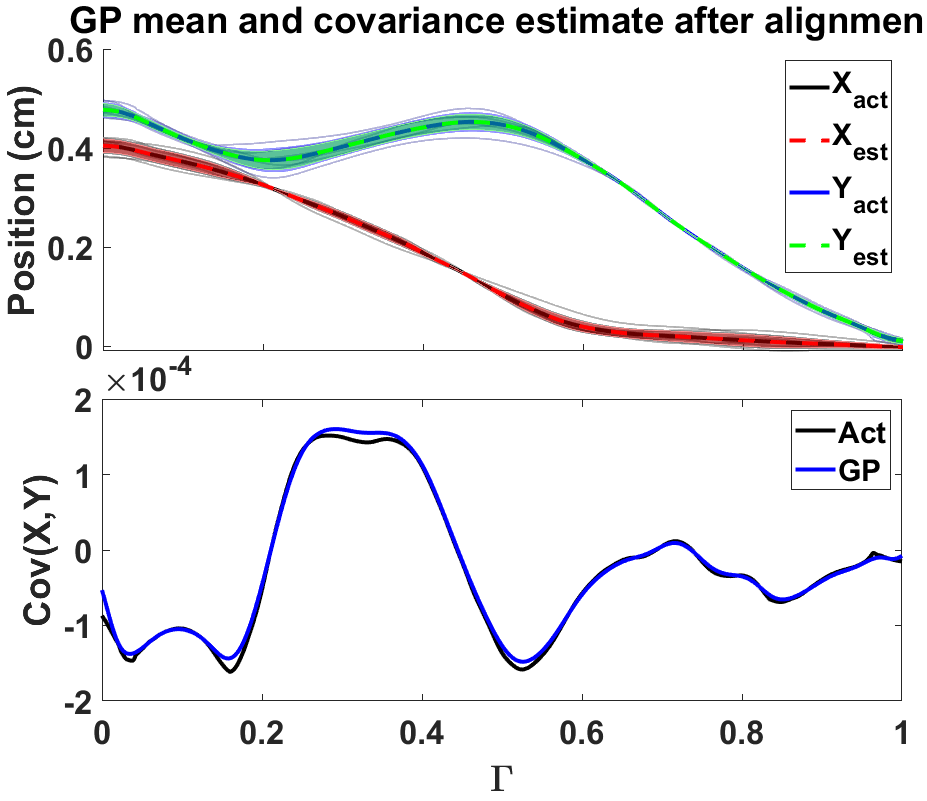


Figure 7.3 FAMGP mean and covariance estimate of the 2D hand written **r** character after phase reparameterization. Top plot shows the mean and variance of X and Y against phase, the estimate almost perfectly overlaps the actual mean and variance at each aligned training sample. Bottom plot shows the actual and GP estimated correlation between X and Y. For this data training was done using 25 eigenvalues and a periodic kernel approximation with a kernel width of 0.1 and period of 1.2 Γ.

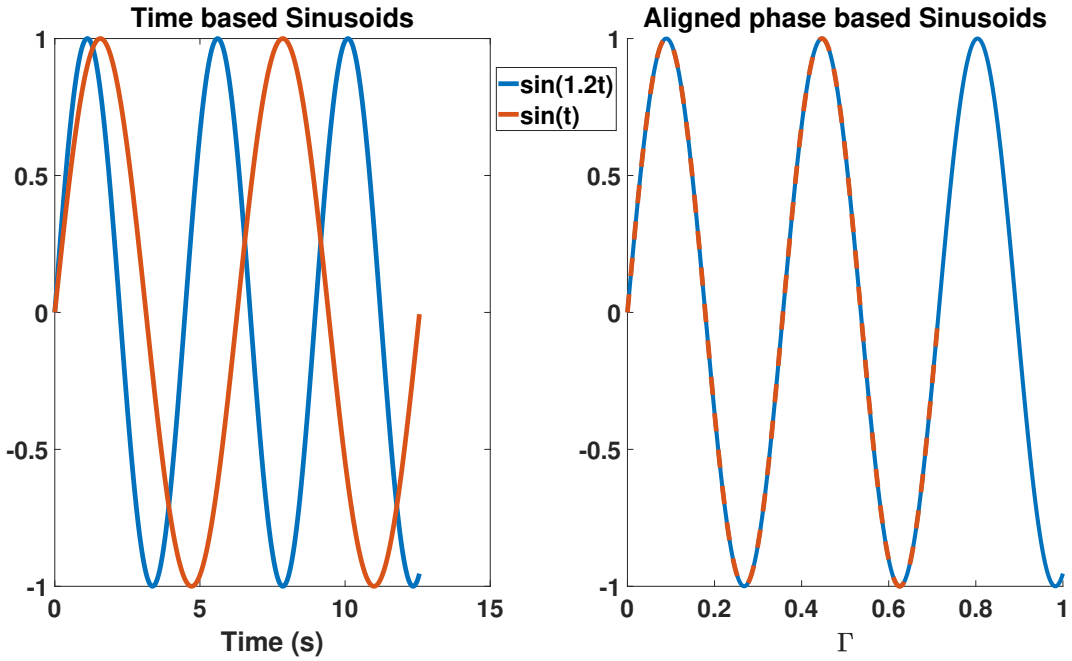


Figure 7.4 Phase parametrization and alignment of sinusoids with different frequencies sampled at 50Hz. Left: time based sinusoid trajectories $sin(t)$ and $sin(1.4t)$. Right: phase parameterized and aligned trajectories. The sampling interval parameter correctly scales the sampling rate to 1.4 times the original to overlap the slower and faster sinusoids. Furthermore, the maximum phase of the slower sinusoid is correctly estimated as 0.7143

7.2 Gaussian Process Model Predictive Control

We propose to utilize the probability density function of a multivariate Gaussian Process trained on exemplar data as the performance index in a linear model predictive control framework. Our formulation allows us to learn the cost function from exemplar trajectories as well as incorporate any additional desired quadratic cost functions. Consider a discrete dynamical system of the form

$$x_k = f(x_{k-1}, u_k) \tag{7.8}$$

We are given multiple exemplar trajectories $[\mathrm{y}_i, t_i]$ provided from expert human demonstrations, where y_i and t_i are the states and sample times for ith exemplar. For example these can be either joint or task space exemplars of human reaching motions we would like to reproduce on a robot. The trajectories are first temporally aligned using the approach described in Section 7.1. From the aligned exemplars we can compute the mean μ_i and covariance Σ_i at each sample. At times when the expert is particularly concerned with a state variable we expect the variance to be low while at a time when the state variable is not important for task completion the variance will be high. Typically, during a reaching motion the human hand will have high variance throughout the middle of the trajectory and low variance towards the end as it approaches the target. Our learned optimal controller should follow the mean trajectory very closely during the low variance regions and can focus on minimizing additional objectives in the high variance regions.

We build a time based Fast Approximate Multivariate Gaussian Process (FAMGP) of the state.

$$x \sim GP(t) \sim \mathcal{N}(\bar{\mu}^*(t), \Sigma^*(t))$$

Where $\bar{\mu}^*(t)$ and $\Sigma^*(t)$ are functions of the exemplar data and current time t. To train this model we consider the mean μ_i at time t_i as output and input respectively, with corresponding covariance Σ_i. Vectorizing all of the means and appropriately stacking the covariance matrices we obtain the training data as described in Section 6.2, where now the vectorized training output is defined as $\mathrm{y} = [\mu_1^1\ \mu_2^1 \cdots \mu_N^1\ \mu_1^1 \cdots \mu_N^2 \cdots \mu_N^M]$ and Σ_{MN} contains the covariances of the training samples. Similar to other stochastic trajectory modeling methods, FAMGP is able to predict the mean and covariance at a specified time. However, it is important to note that the predicted covariance of a Gaussian process typically corresponds to the uncertainty of the prediction and thus decreases with the number of training points near the specified time [177]. Thus, the covariance predicted by the FAMGP model is a smoothed and scaled version of the true output covariance. The scaling factor depends on the selected kernel, hyperparameters, and number of training points. While there is no closed form solution for the scaling factor, upper and lower bounds have been derived for posterior covariance prediction [177, 178]. We will use the covariance prediction

as a quadratic cost of the state in a linear model predictive control (MPC) framework (Section 3.4). In linear MPC only the relationship between the state and input cost functions is important. Thus, we can use the FAMGP covariance prediction and scale the input cost to achieve desired performance.

Discretizing the system with a sampling time of Δt_H, the learned FAMGP can predict μ^* and Σ^* over a discrete time horizon $t^* = [t_0\ t + \Delta t_H\ \cdots\ t_H + \Delta t_H H]$ of length H. Note, the horizon sampling interval Δt_H may be different from the training sampling interval Δt_s or control loop iteration interval Δt_c. This formulation allows us to learn correlations between state variables as well as time samples and efficiently predict both the mean and covariance for a time horizon of t^*. Also, the estimated covariance is guaranteed to be a symmetric positive definite matrix as long as the number of elements in the horizon H does not exceed the selected number of eigenvalues n. We can compute the probability density of the horizon for the entire state as

$$pdf(x_t|T_H) = \frac{e^{-\frac{1}{2}(x_{T_H} - \bar{\mu}^*_{T_H})^T \Sigma^{*-1}_{T_H} (x_{T_H} - \bar{\mu}^*_{T_H})}}{\sqrt{(2\pi)|\Sigma^*_{T_H}|}} \tag{7.9}$$

Notice that the maximum of the pdf over the interval $[t\ t+H\Delta t_H]$ with respect to x is the mean of the Gaussian Process state representation, which closely tracks the mean of the expert exemplars. We view the pdf as a time dependent cost function of the state for a horizon of H samples, where the cost is proportional to the distance to the Gaussian Process mean, weighted by the variance. Furthermore since probability density is positive definite we can instead minimize its negated logarithm. Taking the logarithm and removing terms independent of the state vector horizon x_t

$$-log(pdf(x_{T_H}|t)) \sim \frac{1}{2} x^T_{T_H} \Sigma^{*-1}_{T_H} x_{T_H} - \bar{\mu}^{*T}_{T_H} \Sigma^{*-1}_{T_H} x_{T_H} \tag{7.10}$$

The quadratic form and positive definiteness of the covariance matrix allows us to directly utilize it in the standard constrained linear model predictive control formulation, incorporating any additional performance indexes and placing constraints on the state as well as the control input. Recall that quadratic programming can be used to find the optimal control input $u_{1:N}$ for horizon length N for a discrete linear system of the form

$$x_{k+1} = Ax_k + Bu_k \tag{7.11}$$

$$y_k = Cx_k \tag{7.12}$$

with linear constraints

$$u_{min} \leq P_u u_k \leq u_{max} \tag{7.13}$$

$$x_{min} \leq P_x x_k \leq x_{max} \tag{7.14}$$

Starting at the current state x_0 and propagating over the horizon using (7.11) one can re-write a quadratic cost entirely in terms of the control input

$$J = u_{1:N}^T \underbrace{(\bar{R} + \bar{S}^T \bar{Q} \bar{S})}_{H} u_{1:N} + \underbrace{\vec{x}_0 2 \bar{T}^T \bar{Q} \bar{S}}_{f^T} u_{1:N}. \tag{7.15}$$

Where where $u_{1:N}$ is the optimal control input for the horizon of length N. Matrices $\bar{Q}$ and $\bar{R}$ represent quadratic state and control costs while $\bar{S}$ and $\bar{T}$ are system dynamics over the horizon with respect to current state and the optimal control input [136].

$$\bar{S} = \begin{bmatrix} B & 0 & \cdots & \\ AB & B & 0 \cdots & \\ \vdots & & & \\ A^{N-1}B & A^{N-2}B & \cdots & B \end{bmatrix} \bar{T} = \begin{bmatrix} A \\ A^2 \\ \vdots \\ A^N \end{bmatrix} \tag{7.16}$$

Assuming a sampling interval of Δt we incorporate the Gaussian Processes performance index in addition to other quadratic costs in (7.15)

$$H' = H + \bar{S}^T \Sigma^{*-1}_{T_H} \bar{S} \tag{7.17}$$

$$f'^T = f^T - \bar{S}^T (\bar{\mu}^{*T}_{T_H} \Sigma^{*-1}_{T_H})^T \tag{7.18}$$

The solution of the modified quadratic problem results in model predictive controller that will closely track expert exemplar mean in regions of low variance and focus on minimizing additional costs in areas of high variance.

7.2.1 Application to Manipulators

Our goal is to apply the proposed approach to control a manipulator and reproduce human motion while satisfying the manipulator's joint limits and torque constraints, task space constraints, as well as minimizing additional quadratic cost functions such as control effort. We define the state of the system as the joint $(q,\ \dot{q})$ and end effector $(x,\ \dot{x})$ positions and velocities and linearize the dynamics about the current operating point.

$$\begin{bmatrix} q \\ \dot{q} \\ x \\ \dot{x} \end{bmatrix}_{k+1} = \begin{bmatrix} 1 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & \frac{\Delta t^2}{2} \dot{J} & 1 & \Delta t \\ 0 & \Delta t \dot{J} & 0 & 1 \end{bmatrix} \begin{bmatrix} q \\ \dot{q} \\ x \\ \dot{x} \end{bmatrix}_k + \begin{bmatrix} \frac{\Delta t^2}{2} \\ \Delta t \\ \frac{\Delta t^2}{2} J \\ \Delta t J \end{bmatrix} \ddot{q} \tag{7.19}$$

Where J and $\dot{J}$ are the Jacobian and its derivative and $\ddot{q}$ is the control signal calculated in the inverse dynamics sense [179] using the manipulator's joint torques τ, the inertia matrix M, and the vector of Coriolis, centrifugal, and gravitational terms v at the current state.

$$\ddot{q} = M^{-1}(\tau - \mathrm{v}) \tag{7.20}$$

This formulation allows us to train the GP using either joint trajectories if the kinematics of the manipulator are the same as those of the expert, end effector trajectories, or both. Similarly, constraints and additional cost functions can be in terms of both joint and task space. The linear nature of the inverse dynamics controller allows us to add torque constraints to the system

$$\tau_{min} - \mathrm{v} \leq M\ddot{q} \leq \tau_{max} - \mathrm{v} \tag{7.21}$$

$$\tag{7.22}$$

and quadratic torque cost of $\tau^T R_\tau \tau$ by modifying the performance index

$$H' = H + M^T R_\tau M \tag{7.23}$$

$$f'^T = f^T + M^T R_\tau^T v. \tag{7.24}$$

A useful feature of the proposed approach is that only positional data is needed for training, while the velocity GPs can be computed through GP differentiation using equations 6.13 and 6.14, avoiding numerical differentiation, which is prone to errors, and ensuring that the relative quadratic costs between position and velocity are correct. Also note that since GP estimates a continuous function, the approach does not require the same training and control sampling rates. Thus we can use low sample rate training data in a fast controller without additional data interpolation.

7.2.2 Phase Parameterization

As discussed in Section 7.1, one can re-parameterize the learned GP in terms of phase instead of time and use it in a control application. To apply this within the MPC framework at every iteration of the control loop, first the nearest phase $\Gamma^*(x_0)$ is found using the current manipulator state as a single sample window. Next, the phase is advanced by $\dot{\Gamma}$, a tuning parameter that represents the desired phase velocity. The phase horizon used as input for the GP prediction is thus defined as $\Gamma_H = [\Gamma^*(x_0) + \dot{\Gamma} \ \cdots \Gamma^*(x_0) + \dot{\Gamma} + N\Delta\Gamma_H]$ where $\Delta\Gamma$ is the phase horizon step similar to Δt_h in time based control. The GP is then used to predict the mean and covariance over the phase horizon to be used in MPC. Note that since we normalized the phase variable during training the derivatives predicted by the GP are now with respect to phase and have to be re-scaled by t_1^f.

7.3 Experiments

First, in simulation, we compare the ability of FAMGP to learn a trajectory and its variance to two other popular methods used to encode trajectory distributions, GMM/GMR [90] and ProMP [180]. Next, we extensively validate the proposed approach on the Franka Emika manipulator.

7.3.1 Simulation

We compare the accuracy of modeling time varying mean and covariance of exemplar data of the proposed FAMGP approach with GMM/GMR [90] and ProMP [180] methods. GMM is trained by stacking both the input (time) and output (state) variables together and learns a data representation as K multivariate normal distributions. Next, GMR is used to reproduce the trajectory given new input data, the output mean and covariances are computed conditioned on the known input. The reproduced trajectory is guaranteed to be smooth and the approach can be utilized online since the data is represented through K distributions and thus the output mean and covariance estimation is not dependent on the number of training samples. ProMP represents the mean as a set of weighted squared exponential basis functions, variability is incorporated by placing a Gaussian prior on the weights. After estimating the mean and covariance with either method, MPC can be utilized for control just like with the proposed FAMGP approach.

We learn a simple 2D spiral trajectory that experiences changes in position variance through time.

$$
\begin{aligned}
x &= 0.15tcos(2t) + \epsilon_x \\
y &= 0.15tsin(2t) + \epsilon_y \\
\epsilon_x, \epsilon_y &\sim \mathcal{N}(0, 0.1sin^2(3t))
\end{aligned}
$$

Where $t \in [0, 2\pi]$. 50 trajectories, each containing 200 evenly spaced time points, are used to train the GMM. ProMP, and FAMGP models[1]. To maintain similar prediction computational requirements, the GMM, ProMP, and FAMGP models are trained utilizing 10 mixtures, 10 basis functions, and 10 eigenvalues respectively. We set the period of the kernel to a constant 3π ensuring that the learned model is non repeating within the time span of our training data and allowing for faster training. FAMGP training converges in 1118 gradient descent iterations in 0.47 seconds, GMM means, covariances, and weights are learned in 50.90 seconds using the

[1]Publically available source code for training the GMM and ProMP models was obtained from https://gitlab.idiap.ch/rli/pbdlib-matlab/ and https://github.com/inria-larsen/icubLearningTrajectories respectively, the corresponding publications are [90] and [181]

expectation-maximization algorithm. Since ProMP uses least squares to find the basis functions of each trajectory, it has the fastest training time of 0.27 seconds. It is important to note that the covariance predicted by GPs is scaled based on the selected kernel as well as the number of training data points. More training exemplars will scale down the estimated covariance as the GP model's mean prediction improves [116]. This has no effect on the proposed control methodology since MPC only considers the relative cost between the quadratic state and control performance indexes. In this simulation the FAMGP predicted constant covariance scaling factor is computed as the ratio between the mean ground truth and predicted variances. Figure 7.5 shows the 2D training data and the estimated mean and covariance using the three approaches. Figure 7.6 shows the true and estimated x variances.

The GMM/GMR approach can be viewed as a superposition of K locally linear systems and thus it models the data well near the centers of the 10 Gaussians but lacks accuracy in the transitions. To achieve an accurate model of the spiral mean and covariance one would have to significantly increase the number of mixtures. ProMP computes basis function weights for each exemplar trajectory and calculates the mean and covariance of the weights assuming a Gaussian prior. The covariances of the training data at each time sample are not directly taken into account. On the other hand GP estimation can be viewed as a normal distribution conditioned on every single training time sample mean and covariance. This leads to a better trajectory reproduction with an accurate mean and covariance everywhere. The approximation made by FAMGP allows the application of GP in control by making the estimation independent from training data size while retaining most of the accuracy. It is also important to note that the GP covariance estimate is proportional to the accuracy of the mean estimate and thus is scaled according to the number of training samples. This does not affect the proposed control methodology since MPC treats the cost functions relatively and thus given more or less training data for the GP one has to simply scale the other cost functions accordingly.

7.3.2 Physical Robot Experiments

We extensively test the proposed approach on the Franka Emika manipulator (Fig. 7.7). The robot consists of 7 revolute joints equipped with torque sensors. We use the dynamic model provided by the manufacturer with an addition of a stiction term.

$$\tau = M\ddot{q} + \mathrm{v} + sign(\dot{q})F \tag{7.25}$$

Where $F = [0.75\ 0.98\ 0.50\ 0.97\ 0.89\ 0.25\ 0.45]$ is a constant stiction term found by slowly increasing the joint torque until the joint begins to move.

The manipulator requires 1KHz control rate, using regular GP implemetation utilizing the full training data kernel covariance matrix we were not able to achieve this rate and had to rely

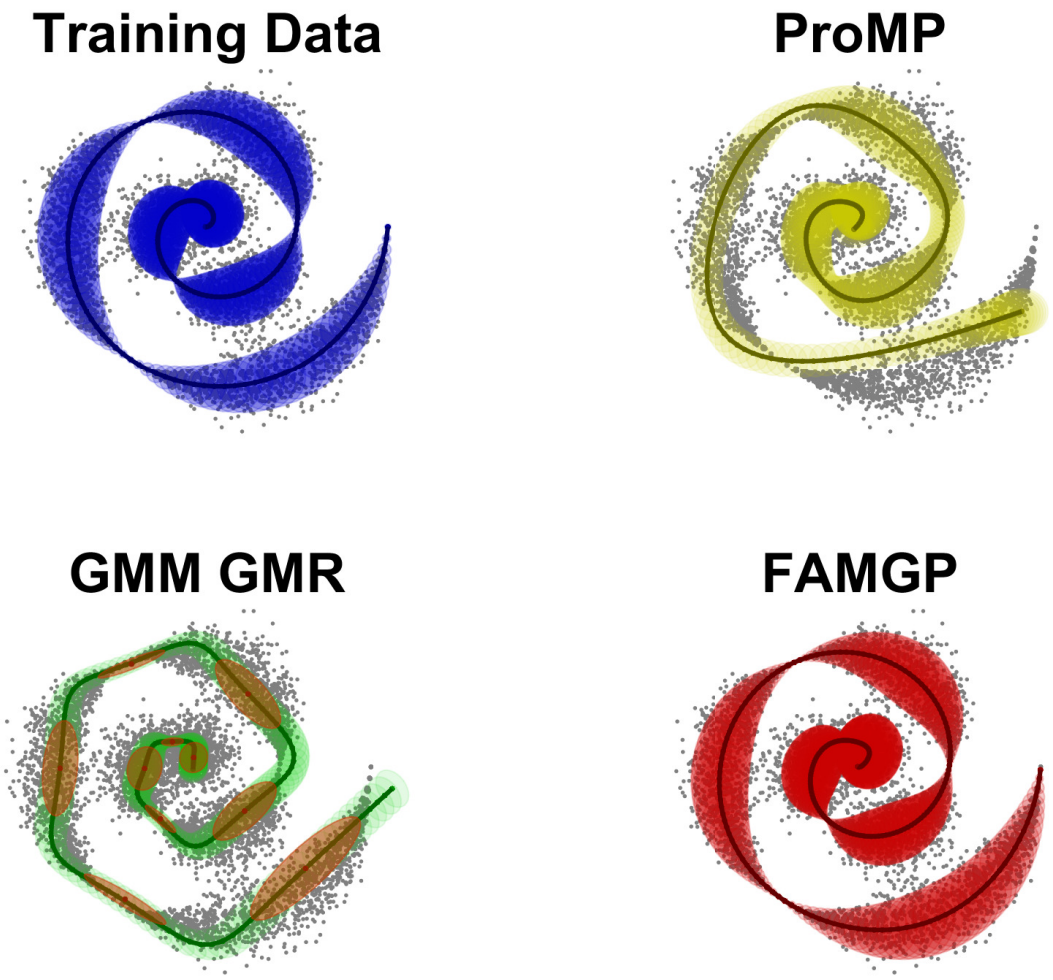


Figure 7.5 Spiral trajectory with changing variance modeling and estimation using the GMM/GMR, ProMP, and FAMGP approaches. GMM optimizes the placement of 10 Gaussians to reflect the exemplar distribution. ProMP estimates basis function weights for each exemplar trajectory and computes their mean and covariance. FAMGP uses the mean and covariance at each timestep projecting it onto a lower dimensional space that can be described by 10 eigenvectors and values.

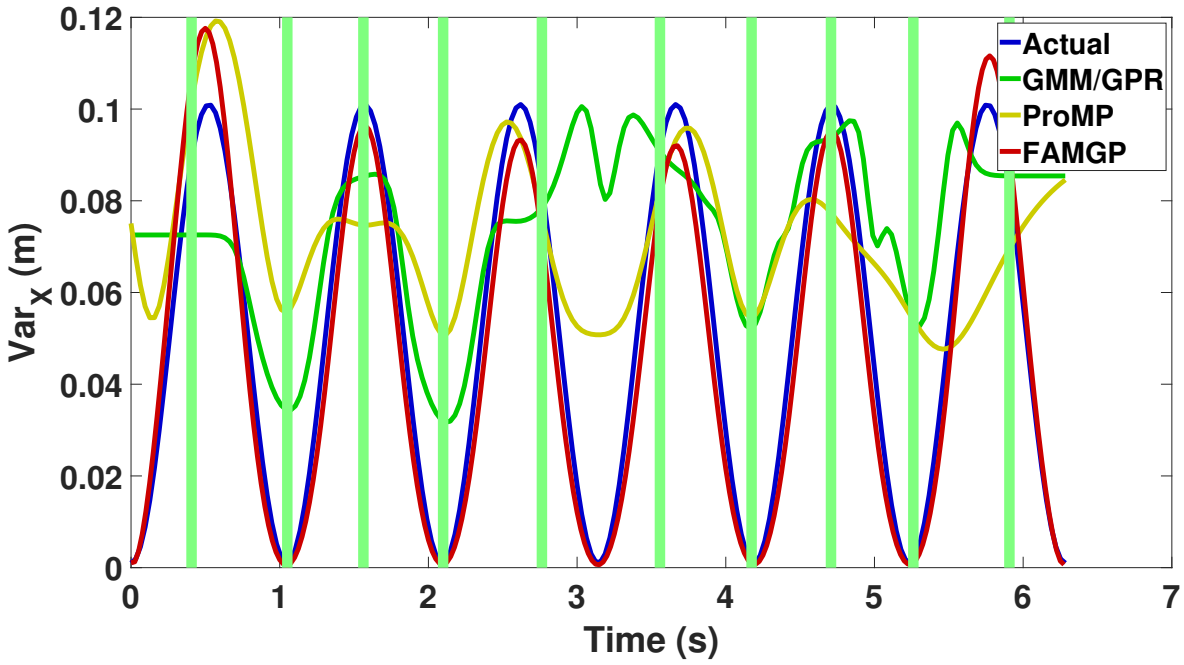


Figure 7.6 GMM/GPR, ProMP, and FAMGP variance estimation. GMM/GPR accurately estimates the variance in regions near the learned Gaussian means (vertical green lines) but does not perform as well in transitional regions. ProMP cannot capture the variance with only 10 basis functions. FAMGP provides an accurate estimate of variance through the entire trajectory.

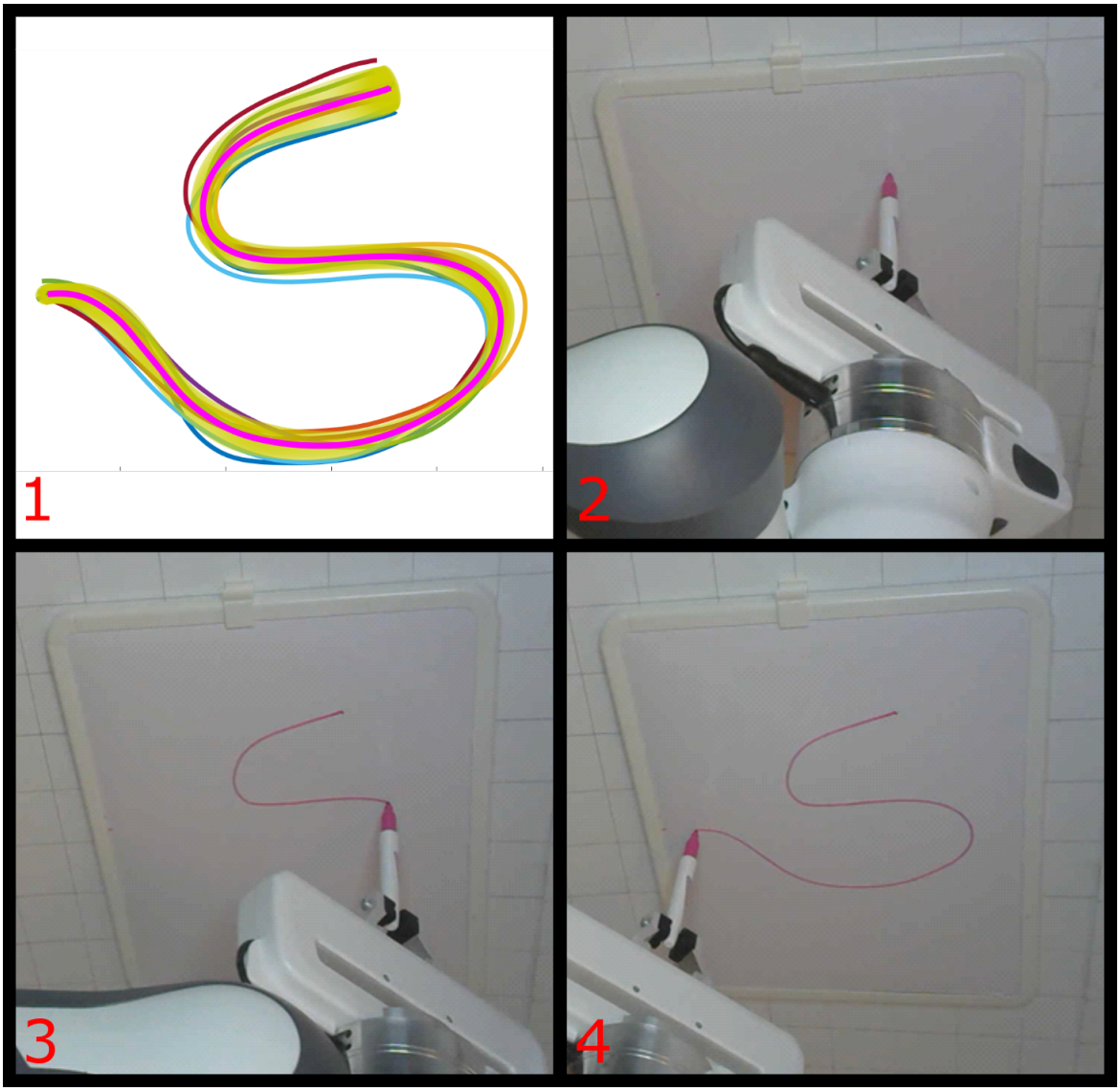


Figure 7.7 Franka Emika 7DOF manipulator drawing the Snake shape learned from human demonstrations. 1: The learned FAMGP model and aligned training exemplar trajectories. 2-4: Beginning, middle, and end of the reproduced motion.

on taking a small block of the matrix about the diagonal. This only works for extremely narrow kernels that quickly fall to zero. FAMGP allowed us to use any size kernel and much more training data while efficiently estimating the mean and covariance over the MPC horizon. To solve the MPC optimization we rely on the Operator Splitting Solver for Quadratic Programs (OSQP) [182]. Utilizing FAMGP and OSQP we were able to formulate and solve the MPC problem with a horizon of 5 steps at the required 1KHz control rate.

7.3.2.1 Parameter Selection

There are multiple tuning parameters both in the FAMGP model as well as the MPC framework. When using the periodic kernel for closed loop trajectories such as the heart shape or the ellipse in Sections 7.3.2.3 and 7.3.2.4, one can set the period exactly to the temporal length of the training data to ensure smooth repetitions. When repetitions are not required the period should be longer than the length of the entire motion. Choosing the kernel width is also important, as the kernel gets narrower a larger number of eigenvalues is required to accurately approximate a GP, which increases the regression computational complexity. For all of the robot experiments we used a kernel of width 0.1 with 25 eigenvalues, which capture 97% of the data. With these settings we were able to predict a horizon of 5 steps and compute the optimal torques at 1KHz. In the MPC framework one must select a horizon step length as well as the control input quadratic cost. Since the true system dynamics are non-linear the system would not be accurately approximated with the linearization for a large horizon. A very short horizon can lead to oscillations about the mean of the FAMGP model. For the robot experiments the horizon step length was set to $t_H = 0.1$ leading to a look-ahead of 0.5 seconds. In addition to the specified constraints of each experiment we incorporate a constant small cost on joint accelerations of 0.01 and a constraint on the change of acceleration $\ddot{q} < 3750$ in order to minimize joint torques switching sign and causing vibrations.

7.3.2.2 Correlation Learning

First, we demonstrate the benefits of learning a correlated cost function using the multioutput GP. We train a static task space GP to hold the end effector at (0.4 0.1 0.2) with zero roll pitch and yaw. In one scenario Σ_N is set such that the GP outputs are independent with a variance of 1cm and 0.01 rad for position and orientation of the end effector respectively. In the other, x and z position are highly correlated at 90%. Here we utilize a time based GP with a period of 7 seconds. The training data consists of 1000 equally spaced samples.

We simulate a 40Nm disturbance applied to the end effector in the x direction for 1 seconds by adding a torque $\tau_d = J^T [40\ 0\ 0\ 0\ 0\ 0]^T$ to the control signal at the 1 second mark. Fig. 7.8

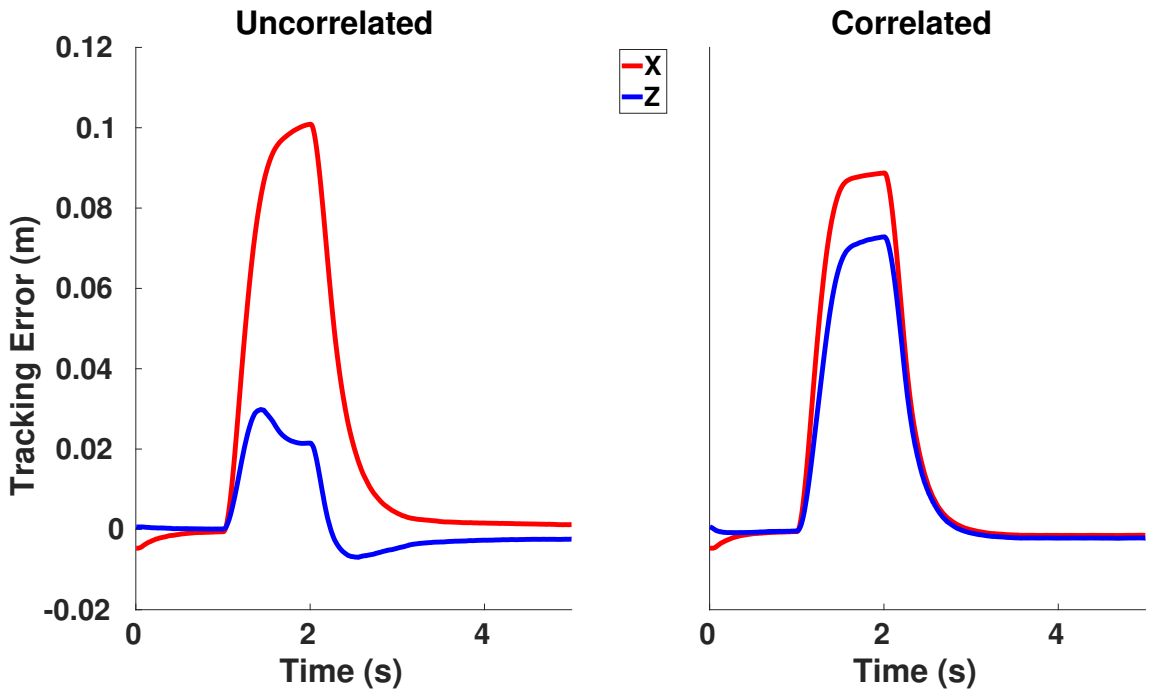


Figure 7.8 Uncorrelated and correlated task space end effector tracking during a disturbance.

shows the end effector position tracking accuracy for the uncorrelated and correlated scenarios. When the positions are uncorrelated a disturbance in the x direction causes the end effector to also move predominantly in the x direction. The error in z position tracking is the result of the controller minimizing roll pitch and yaw as well as the position simultaneously. When the x and z positions are highly correlated and the end effector begins experiencing motion in the x direction due to the applied force, the controller maintains the correlation and applies torques such that the end effector also experiences motion in the z direction. The slightly better tracking towards the end of the trajectory in the correlated case can be attributed to the fact that adding the off diagonal correlation elements increases the matrix 2 norm of the covariance matrix inverse, leading to the MPC controller focusing on tracking x and z positions.

7.3.2.3 Task Space Constraints

Next we demonstrate the ability of the proposed method to learn and track a trajectory while satisfying constraints. For this experiment the robot is tasked with tracing out a shape on a 3d plane in the task space. We take 500 evenly spaced samples of a heart shaped trajectory using

the equation

$$x_h = \theta_h . * sin(\frac{\pi * .872 * sin(\theta_h)}{\theta_h})$$
$$y_h = -abs(\theta_h) . * cos(\frac{\pi * sin(\theta_h)}{\theta_h})$$

where $\theta_h \in (-\pi, \pi)$. To increase the tracking difficulty we rotate and move the points in 3 dimensions resulting in a 3d heart outline trajectory. A FAMGP is trained using the points, we assume uncorrelated variance of 1 cm and and 5 degrees for the position and orientation dimensions respectively. The orientation is assumed constant to keep the end effector perpendicular to the heart outline in 3d space. Periodic kernel with a period of 10 seconds ensures that the manipulator will continuously repeat the trajectory. Fig 7.9 shows the tracking performance of the proposed approach as well as its ability to handle an additional position constraint $x < 0.525$. Fig. 7.10 shows the tracking error with respect to time over 6 repetitions of the unconstrained trajectory.

The proposed method successfully tracks the learned trajectory while satisfying a task space constraint. Ideally we would expect perfect tracking, however due to small discrepancies between our dynamic model and the real manipulator we observe small errors in the end effector position and orientation of 1 cm and 4 degrees respectively. The largest contributor to the error is our simplistic friction model. When the end effector is very close to the GP mean, the optimal joint accelerations calculated by MPC are very small and the computed torques may not be enough to actually generate the desired accelerations. One can increase the optimal accelerations by reducing the joint acceleration cost, however due to unmodeled dynamics this may cause vibrations as the manipulator very quickly oscillates about the GP mean.

7.3.2.4 Phase Parameterization

To demonstrate the benefits of the proposed phase trajectory re-parameterization control application (Sec 7.2.2) the robot is tasked to track an ellipse on the x-y plane using either the proposed time or phase parameterized control. Similar to the previous experiment, the FAMGP model is trained with 500 evenly spaced samples using a periodic kernel such that the ellipse is continuously repeated with a period of 10 seconds. To simulate the manipulator encountering an obstacle, after one ellipse repetition we set the torque to only compensate for gravity for 5 seconds, effectively stopping any end effector motion. In order to achieve similar end effector velocities for time and phase parametrization $\dot{\Gamma}$ was tuned to be 0.14. Figure 7.11 shows the MPC torque outputs from the phase and time parameterized controllers for two joints that are mainly

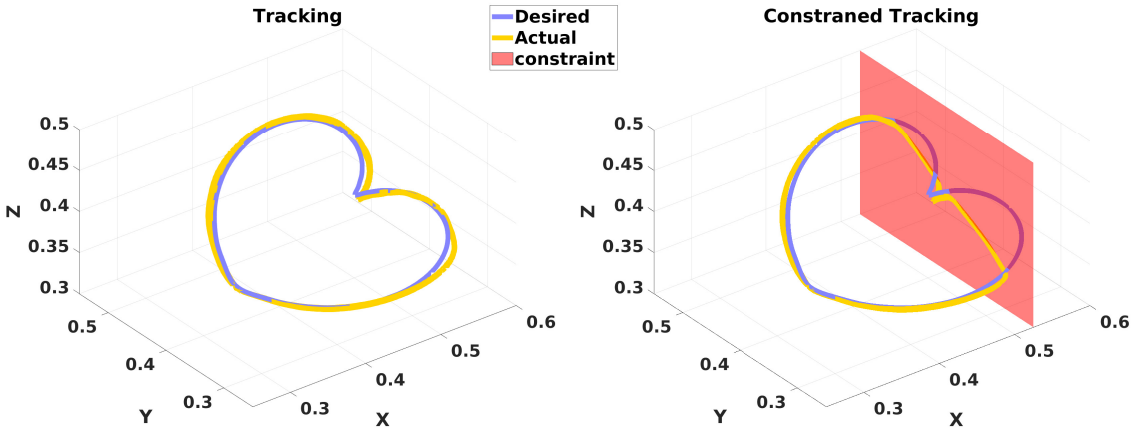


Figure 7.9 Unconstrained (left) and constrained (right) end effector tracking of a heart outline in 3d space. Without a constraint the manipulator accurately tracks the learned trajectory. Once a task space constraint is introduced the proposed control approach ensures that it is satisfied.

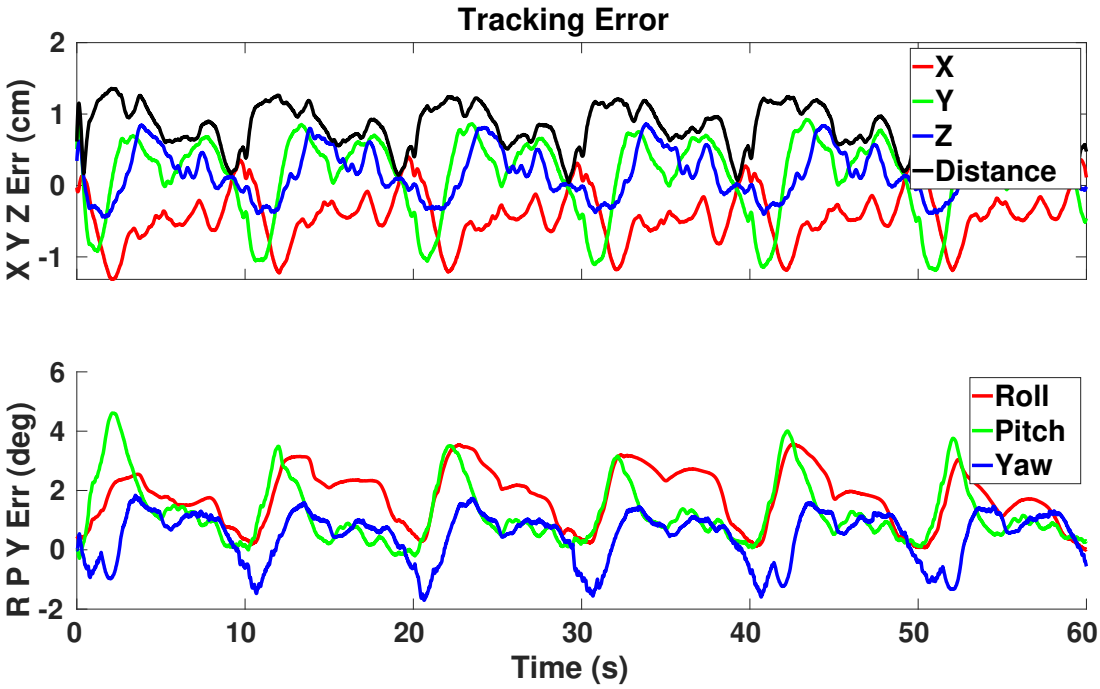


Figure 7.10 Unconstrained heart outline position and orientation tracking errors over 6 trajectory repetitions.

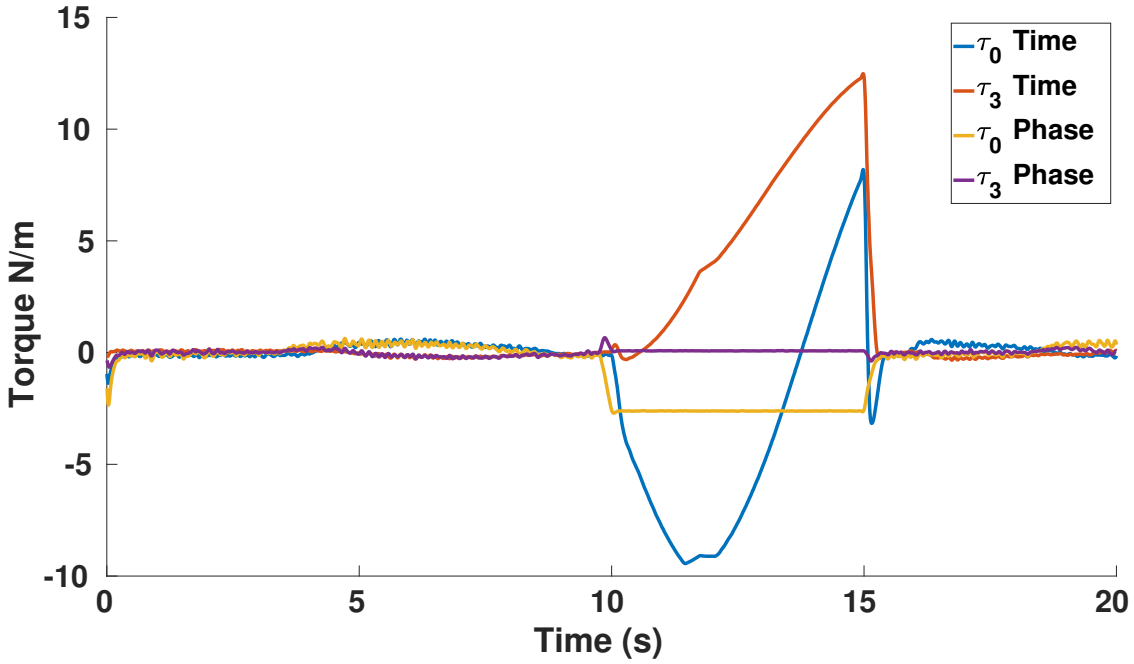


Figure 7.11 Torques of the two joints responsible for end effector motion in the x-y plane for time and phase parameterized MPC controllers.

responsible for end effector motion in the x-y plane. Figure 7.12 shows the trajectory tracking of the time and phase parameterized controllers.

When at the 10 second mark the manipulator encounters an obstacle, the time parametrized controller continues to increase the joint torques as the FAMGP predicted end effector position gets further and further away. After the obstacle is removed the high torques quickly allow the manipulator to converge to the elliptic trajectory. However, since the manipulator was stopped for 5 seconds and the trajectory period is 10 seconds, the predicted end effector position is now on the other side of the ellipse, thus the time parameterized controller allows the end effector to cut through the middle of the ellipse. The phase based controller finds the nearest phase at each loop iteration and this does not change while the obstacle prevents the manipulator from moving. MPC only increases the torques enough to try and achieve desired end effector velocity at the nearest phase. After the obstacle is removed the manipulator smoothly continues to track the ellipse.

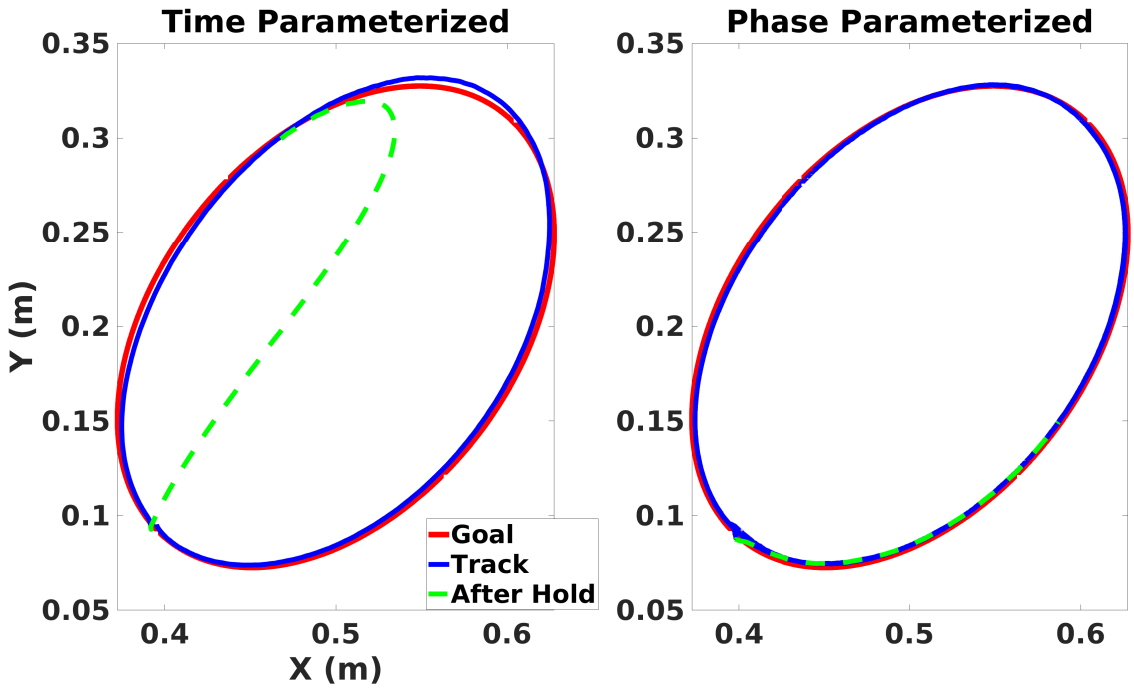


Figure 7.12 End effector tracking of the elliptical trajectory for time and phase parameterized MPC. The green dashed line shows the end effector position for 2 seconds after the obstacle is removed.

7.3.2.5 LASA Dataset

Finally we demonstrate the capabilities of the proposed approach to learn and reproduce real human motions. We use the LASA handwriting dataset [176] consisting of 7 demonstrations of 26 different 2D patterns collected on a tablet. First we use the proposed phase parameterization approach (Sec 7.1) to time align the trajectories and ensure each exemplar is 5 seconds long allowing for safe end effector velocities. After alignment we translate and scale the data so that all the trajectories are in the reachable space of the manipulator and calculate the x, y mean and covariance at each time sample. For each of the character patterns in the dataset, an FAMGP model (Chap 6) is trained using the x, y mean and covariance and assuming constant end effector height of 26cm with and zero roll pitch and yaw with variances of 1cm and 1 degree respectively. This allows the manipulator to hold a paintbrush and draw the patterns on a flat surface. The period of the kernel is set to 7 seconds in order to make sure the entire trajectory is learned without repetition. Figure 7.13 shows the entire process and tracking results for one of the patterns. Table 7.1 provides the tracking accuracy over all of the 26 patterns.

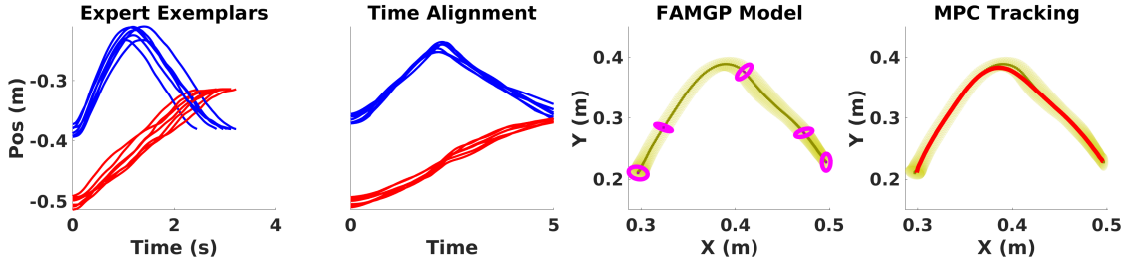


Figure 7.13 Proposed approach applied to hand writing dataset using the Panda manipulator. From left to right, 1) Original Demonstrations X red, Y blue: Since the demonstrations were performed at inconsistent velocities directly computing the mean and covariance at each time sample would lead to an inaccurate position distribution representation. 2) Time Alignment (Sec 7.1): Using the proposed approach the demonstrations are aligned to a consistent time variable. 3) FAMGP Modeling (Chap 6): FAMGP model of the hand-written character is learned using the aligned mean and covariance, to fully capture the task space the FAMGP also models a constant end effector height of 0.26m with variance of 1cm and zero in roll, pitch, and yaw with variance of 5 degrees. The purple ellipses show the FAMGP covariance estimates at selected times. 4) MPC Tracking (Sec 7.1.1): Model Predictive Control traces the character looking ahead 0.5 seconds with a 5 sample horizon. The mean distance to the desired X and Y position is 0.24cm while roll, pitch, yaw, and Z position are tracked with an accuracy of 0.25, 0.28, 1.02 degrees and 4mm.

Table 7.1 Full Lasa data-set tracking performance. Since the patterns are traced by the manipulator on the X-Y plane the X-Y distance is shown separately from the end effector Z position error.

XY Dist (cm)	**Z Err (cm)**	**Roll (Deg)**	**Pitch (Deg)**	**Yaw (Deg)**
0.30 ± 0.21	0.4 ± 0.3	0.43 ± 0.33	0.44 ± 0.24	0.96 ± 0.54

7.4 Summary

This chapter proposed a novel approach to transfer expert skills onto a manipulator. Exemplars of motion are aligned in time and are used to learn a fast approximate Gaussian process (FAMGP) model that captures the mean and covariance of the movement. At each iteration of the control loop the FAMGP predicts the mean and covariance over a horizon and this prediction is used as to form a quadratic cost function within a model predictive control framework, allowing us to incorporate constraints or additional quadratic costs. Thus, in regions where expert exemplars had high variability the controller will focus on minimizing control inputs and additional costs, in low variability regions it will precisely track the expert exemplar mean. Re-parameterizing the FAMGP model in terms of a phase variable and at each iteration finding the nearest phase based on the current state allows the manipulator to handle encountering obstacles while always maintaining the desired trajectory. We compare the approach in simulation to Gaussian mixture modelling showing that FAMGP better captures the mean and covariance using the same number of parameters. Extensive testing on a real manipulator demonstrates correlation learning, satisfying constraints, phase re-parameterization, and reproducing real hand drawn patterns.

Chapter 8

Conclusions and Future Work

For millennia people have dreamt of creating intelligent machines on par with their human creators. Ancient Greeks developed mechanical automatons powered by steam and water using complex gear mechanisms to achieve the simplest of motions. Fiction writers imagined humanoids faster, stronger, and smarter than their human counterparts. Robots have always enticed our curiosity, and almost always the ultimate form was a machine created in our own image. As technology advanced the robotic devices surpassed our physical abilities, assembly arms lift heavy car parts and bolt them on with incredible precision, pick and place machines accurately position hundreds of tiny electronic components per minute. However, most robots today are single task oriented, move in a predefined trajectory, and do not react to disturbances. This type of motion is not conducive to human robot collaboration both due to safety concerns and the fact that in collaborative tasks a lot of human communication is non-verbal [11]. If we want robots to truly blend into our environment we need their movements to be human like, capturing the underlying control strategies we use for specific tasks. This would allow the devices to convey intent though motion, react to disturbances in a predictable human like manner, and provide a layer of safety during the interaction. The work presented in this thesis is a step towards designing such controllers. To achieve this we first develop algorithms to capture human motion using different modalities. Then, we create a novel learning from demonstration approach capable of extracting the control strategy used by participants from motion trajectory exemplars. This chapter summarizes the work presented in this thesis and presents future research directions.

8.1 Summary

In Chapter 4 we demonstrate how human motion is better represented on Lie groups instead of traditional Euler angle joint modelling. Lie group Kalman filter is derived to estimate movements using motion capture markers or wearable sensors. We show that the proposed method improves estimation accuracy because it is not effected by gimbal lock and can handle lower sampling rates. Chapter 5 shows how incorporating constraints into the Kalman filter can improve pose estimation accuracy. We also propose a new method to determine the most likely active constraint which allows us to track global body position during gait with only wearable IMUs. Both of the motion estimation algorithms are extensively validated with real human motion experiments.

To learn controllers from demonstrations and re-target to robotic devices we aim to capture the variability and interdependence between states in the expert motion. Multivariate Gaussian processes are well suited for this application as they are smooth function estimators that capture both the mean and variance of the data, as well as the correlations between the outputs. However, Gaussian process regression is dependent on the number of training points and thus is difficult to utilize in real time control. In Chapter 6, a novel fast approximate multi-output Gaussian process that approximates the kernel function using a selected number of eigen values is developed. With the proposed approach, regression is independent from the number of training points and can be used in real time applications to estimate mean and covariance. Furthermore, model training has exponential complexity with respect to the number of data points in a regular Gaussian process formulation. In case of the proposed approximation, training complexity is linear and in a special case independent from the size of the data set.

In Chapter 7 we show how the Gaussian process approximation can be used as a learned, unparametric, time varying cost function in a model predictive control framework. This leads to a controller that generates motion and reacts to disturbances in a human like manner. Furthermore, it can satisfy task, joint space, or torque constraints handling obstacles and ensuring safety in collaborative tasks. The proposed approach is shown to outperform other stochastic trajectory modeling methods in simulation and is extensively validated on a 7DOF manipulator utilizing real human motion exemplars.

8.2 Future Work

This section describes possible future research directions for the various contributions presented in the thesis.

8.2.1 Human Motion Estimation Future Directions

In Chapters 4 and 5 we presented different approaches to estimate human motion using motion capture markers or wearable sensors. Both the proposed methods and the majority of other state of the art approaches assume that the placement of the markers or IMUs is known. However, our observation analysis in Chapter 4 showed that the joint angles can become unobservable under certain marker placement conditions. Extending the observability analysis to include IMU based measurements can help understand the benefits and limitations of wearable sensor placement. This can lead to a better understanding of how many IMUs are required to accurately estimate full body motion and which segments they should be worn on.

In Chapter 5 we demonstrated that adding constraints into pose estimation can greatly improve the accuracy. While the proposed approach can simultaneously enforce multiple active constraints, it is only capable of selecting a single most likely active constraint from a set. Future work will focus on detecting the situation when a subset of more than one constraint is active without needing to implement an EKF model for each constraint case. This will allow for a double support stage during gait and can further improve the estimation accuracy. Beyond the gait application it would enable arbitrary interactions with the environment. Finally, in order to achieve near real time performance, the constraints presented in this chapter were derived assuming a prismatic and revolute joints instead of Lie group elements. However, the Lie group Kalman filter and Jacobians presented in Chapter 4 can also be used and should improve the estimation accuracy, particularly in near gimbal lock regions.

8.2.2 Fast Approximate Multivariate Gaussian Processes Extensions

The proposed Gaussian process approximation allows for very efficient training and regression with non-parametric models. The computational complexity and accuracy of the approximation is directly related to the chosen number of eigen values. Currently this can be considered a tuning parameter of the algorithm, future work will include automatically increasing or reducing the number of eigenvalues during hyperparameter optimization by considering the ratio between the largest and smallest. This will allow training to speed up for wider kernels and maintain accuracy for very narrow ones. We also want to explore the applicability of the kernel approximations to multiple inputs, combining multiple kernels, and exploring additional available Mercer expansions. This would allow for learning much more complex processes. Finally, it may be possible to further optimize GP training and regression by combining the proposed approach with existing inducing points methods [121, 124] leading to Gaussian processes capable of handling extremely large datasets.

8.2.3 Learning from Demonstration

In Chapter 4 we showed how human motion and its variance is better represented on Lie groups. While the representation of each joint depends on its degrees of freedom, the end effector position and orientation can always be described by an element of SE(3). However, when implementing the proposed objective function learning method, described in Chapter 7, we had to revert back to Euclidean coordinates for the end effector position and orientation representation since neither the multivariate Gaussian processes nor the necessary Jacobian time derivatives are well defined for end effector SE(3) representation. Maintaining a consistent SE(3) representation between capturing the motion, learning the objective function, and controlling the robot may lead to an improvement in human like movement of the manipulator. For this we need a re-formulation of the FAMGP model and manipulator Jacobians.

8.2.3.1 Tangent Space Objective Function

Consider M temporally aligned end effector trajectory exemplars of a particular task $[\mathbf{Y}_i, \mathbf{t}]$ where $\mathbf{Y}_i = [\mathbf{y}_1^i,\ \mathbf{y}_2^i, \cdots,\ \mathbf{y}_N^i]^T$ are the i_{th} exemplar trajectory end effector poses in some representation and $\mathbf{t}$ is a vector of N sample times. In Chapter 5, in order to train the approximate Gaussian process model, at each time sample we computed the mean and variance of the end effector pose using the Cartesian position and roll, pitch, yaw representation. Instead we can represent each sample as an element of SE(3).

$$\mathbf{y}_k^i = \begin{bmatrix} R_k^i & r_k^i \\ 0 & 1 \end{bmatrix} \in \mathrm{SE}(3)$$

At every time sample k we can iteratively find the mean $\mu_k \in \mathrm{SE}(3)$ and variance $\mathbf{\Sigma}_k \in \mathbb{R}^{6\times 6}$ from the exemplar trajectories. First, an initial mean is chosen as one of the samples, next the new mean and covariance are computed in terms of deviations on the tangent space [183].

Algorithm 2 SE(3) Sample Mean and Covariance [183]

Result: $\mu_k,\ \mathbf{\Sigma}_k$

$\mu_k = \mathbf{y_k^1}$

for j < *number of iterations* **do**

  $\mathbf{v^m} \equiv \log_{\mathrm{SE}(3)}(\mathbf{y}_k^m \mu_k^{-1})^{\vee}_{\mathrm{SE}(3)} \in \mathbb{R}^6,\ m \in [1 \cdots M]$

  $\mathbf{\Sigma}_k \leftarrow \frac{1}{M}\sum_{m=1}^{M} \mathbf{v^m}\mathbf{v^m}^T$

  $\mu_k \leftarrow \exp_{\mathrm{SE}(3)}(\frac{1}{M}\sum_{m=1}^{M} \mathbf{v^m}^{\wedge}_{\mathrm{SE}(3)})\mu_k$

end

We can then train a time based FAMGP model using the $[\log_{\mathrm{SE}(3)}(\mu_k)]^{\vee}_{\mathrm{SE}(3)} \in \mathbb{R}^6$ mean and $\mathbf{\Sigma}_k \in R^{6\times6}$ variance samples $GP(t) \sim \mathcal{N}(\log_{\mathrm{SE}(3)}(\bar{\mu}(t))^{\vee}_{\mathrm{SE}(3)}, \mathbf{\Sigma}(t))$.

To use the model in a model predictive control framework we must define a quadratic cost utilizing the FAMGP model. Similarly to the learning controller presented in Chapter 7, we rely on the negative logarithm of the probability, however this time the distance between the manipulator end effector and the FAMGP model mean is defined as a deviation on the tangent space of SE(3). Consider the current true end effector pose $\mathcal{K}^0_{ee}(\mathbf{q}_t) \in \mathrm{SE}(3)$ in terms of the joint angles $\mathbf{q}_t$, the deviation $\mathbf{d} \in \mathbb{R}^6$ from the mean at time t is defined as $\mathbf{d} = \log_{\mathrm{SE}(3)}(\mathcal{K}^0_{ee}(\mathbf{q})\bar{\mu}(t)^{-1})^{\vee}_{\mathrm{SE}(3)}$ and thus we formulate a quadratic objective function using the negative log of the probability of being in the current pose.

$$-log(pdf(\mathcal{K}^0_{ee}(\mathbf{q}_t)|t)) \sim \frac{1}{2}\mathbf{d}^T\mathbf{\Sigma}(t)\mathbf{d} \tag{8.1}$$

We can also use the same approach to learn objective functions for relative movements. For example we can learn how to hand objects in a human-like manner. Consider multiple exemplar trajectories of a task where one participant hands a tool to another. Instead of training a FAMGP model on the hand pose of one of the participants we can train it on the transformation between the two end effectors in the frame of the giver. The learned model will have high variance at the beginning of the trajectory since at the start of the handover is not constrained in any way. At the end of the trajectory the end effectors must meet for a successful handover and thus the transformation between them must get close to identity and zero deviation on the tangent space. This will ensure that our model will have a low variance towards the end of the trajectory. In this sense, designing a controller that drives the deviation to zero will ensure a successful handover.

To incorporate this objective function into a model predictive control framework we require the Jacobian and its time derivative of the deviation $\mathbf{d}$ with respect to the joint angles of the manipulator. The next section provides an outline for the analytical differentiation.

8.2.3.2 Tangent Space Jacobians

To utilize the cost function defined on the tangent space in the model predictive control framework presented in Chapter 7, we required the Jacobian and its time derivative of the tangent space deviation with respect to the joint angles of the manipulator. Recall that the forward kinematics of the robot can be decomposed into a combination of transformations up to a specific joint l from world frame (0), transformation due to the $l_t h$ joint, and transformation after the joint to the end effector (ee) (equation 4.15).

$$\mathcal{K}^0_{ee}(\mathbf{q}_t) = \mathcal{K}^0_l \theta_l \mathcal{K}^{l+1}_{ee} \tag{8.2}$$

where $\theta_l = \exp_{SE(3)}(q_l E^z) \in \mathrm{SE}(3)$ represents the rotation about the z axis of q_l radians and E^z is the constant z axis generator. Using the basic matrix exponential property $\exp(YXY^{-1}) = Y\exp(X)Y^{-1}$ for any invertible Y we can re-write the forward kinematics equation, bringing the effect of the l_{th} joint to the left side.

$$\mathcal{K}^0_{ee}(\mathbf{q}_t) = \exp_{SE(3)}(q_l \mathcal{K}^0_l E^z \mathcal{K}^l_0)\mathcal{K}^0_l \mathcal{K}^{l+1}_{ee} \tag{8.3}$$

and the deviation written in terms of the l_{th} joint becomes

$$\mathbf{d} = \log_{\mathrm{SE}(3)}\Big(\exp_{SE(3)}(q_l \mathcal{K}^0_l E^z \mathcal{K}^l_0)\underbrace{\mathcal{K}^0_l \mathcal{K}^{l+1}_{ee}\bar{\mu}^{-1}(t)}_{K'_l}\Big)^{\vee}_{\mathrm{SE}(3)} \tag{8.4}$$

Where $K'_l \in \mathrm{SE}(3)$ is the contribution to the deviation from all other joints and the GP mean. Letting $u_l = \mathcal{K}^0_l E^z \mathcal{K}^l_0 \in \mathfrak{se}(3)$ and $v_l = \log_{\mathrm{SE}(3)}(K'_l) \in \mathfrak{se}(3)$ we achieve an expression of the deviation in terms of the l_{th} joint as the logarithm of a product of two exponentials.

$$\mathbf{d} = \log_{\mathrm{SE}(3)}(\exp_{SE(3)}(q_l u_l)\exp_{SE(3)}(v_l))^{\vee}_{\mathrm{SE}(3)} \tag{8.5}$$

Typically one would have to use the Baker–Campbell–Hausdorf expansion formula which expresses such a product as an infinite series. However, for the special case of SE(3) there is a closed form solution [184]. Splitting the deviation and each exponential into the translational $\mathbf{d}_t, u_{l,t}, v_{l,t} \in \mathbb{R}^3$ and rotational $\mathbf{d}_{\phi^{\wedge}_{\mathrm{SO}(3)}}, u_{l,\phi}, v_{l,\phi} \in \mathfrak{so}(3)$ components where $\mathbf{d} = [\mathbf{d}_t^T \mathbf{d}_\phi{}^T]^T$ and

$$u_l = \begin{bmatrix} u_{l,\phi} \in \mathfrak{so}(3) & u_{l,t} \\ \mathbf{0} & 0 \end{bmatrix},\ v_l = \begin{bmatrix} v_{l,\phi} \in \mathfrak{so}(3) & v_{l,t} \\ \mathbf{0} & 0 \end{bmatrix} \in \mathfrak{se}(3)$$

we can compute it as

$$\mathbf{d}^{\wedge}_{\mathrm{SE}(3)} = \begin{bmatrix} \mathbf{d}_{\phi^{\wedge}_{\mathrm{SO}(3)}} \in \mathfrak{so}(3) & \mathbf{d}_t \\ \mathbf{0} & 0 \end{bmatrix} \tag{8.6}$$

where

$$\mathbf{d}_\phi = \alpha_1 q_l u_{l,\phi^{\vee}_{\mathrm{SO}(3)}} + \alpha_2 v_{l,\phi^{\vee}_{\mathrm{SO}(3)}} + q_l \alpha_{12}(u_{l,\phi^{\vee}_{\mathrm{SO}(3)}} \times v_{l,\phi^{\vee}_{\mathrm{SO}(3)}}) \tag{8.7}$$

$$\mathbf{d}_t = T_1 q_l u_{l,t} + T_2 v_{l,t} \tag{8.8}$$

where the scalars $\alpha_1,\ \alpha_2,\ \alpha_{12}$ and invertible tensors T_1 and T_2 can be found in closed form. Letting $w_1 = |q_l u_{l,\phi^\vee_{SO(3)}}|$ and $w_2 = |v_{l,\phi^\vee_{SO(3)}}|$

$$\alpha_1 = \frac{sinc(\frac{w_1}{2})cos(\frac{w_2}{2})}{sinc(\frac{w}{2})} \tag{8.9}$$

$$\alpha_2 = \frac{cos(\frac{w_1}{2})sinc(\frac{w_2}{2})}{sinc(\frac{w}{2})} \tag{8.10}$$

$$\alpha_{12} = \frac{sinc(\frac{w_1}{2})sinc(\frac{w_2}{2})}{2sinc(\frac{w}{2})} \tag{8.11}$$

$$w = 2cos^{-1}\left(cos(\frac{w_1}{2})cos(\frac{w_2}{2}) - \frac{1}{4}sinc(\frac{w_1}{2})sinc(\frac{w_2}{2})w_1 w_2\right) \tag{8.12}$$

$$T_1 = \partial \exp(\mathbf{d}_{\phi^\wedge_{SO(3)}})^{-1} \partial \exp(q_l u_{l,\phi}) \tag{8.13}$$

$$T_2 = \partial \exp(\mathbf{d}_{\phi^\wedge_{SO(3)}})^{-1} \exp(q_l u_{l,\phi}) \partial \exp(v_{l,\phi}) \tag{8.14}$$

$$\tag{8.15}$$

Where for $\phi \in \mathfrak{so}(3)$ $\partial \exp(\phi)$ and $\partial \exp(\phi)^{-1}$ are defined as follows [184]:

$$\partial \exp(\phi) = \mathbf{I}^3 + \frac{1}{2}sinc^2\left(\frac{|\phi^\vee_{SO(3)}|}{2}\right)\phi + \left(1 - sinc(|\phi^\vee_{SO(3)}|)\right)\frac{\phi^2}{|\phi^\vee_{SO(3)}|^2} \tag{8.16}$$

$$\partial \exp(\phi)^{-1} = \mathbf{I}^3 - \frac{1}{2}\phi + \left(1 - \frac{|\phi^\vee_{SO(3)}|}{2}cot\left(\frac{|\phi^\vee_{SO(3)}|}{2}\right)\right)\frac{\phi^2}{|\phi^\vee_{SO(3)}|^2} \tag{8.17}$$

Thus we have a closed form expression for the deviation contribution due to the l_{th} joint defined entirely on the tangent space. All of the terms in the above expression are differentiable with respect to q_l and therefore we can compute the required derivatives $\frac{\partial \mathbf{d}}{\partial q_l}$. Iterating over all of the joints the full tangent space Jacobian can be built. Finally, to use the tangent space representation in our model predictive control framework we also require the temporal derivative of the Jacobian which can be computed by applying chain rule to the Jacobian $\dot{J} = \frac{\partial J}{\partial t} = \frac{\partial J}{\partial \mathbf{q}}\frac{\partial \mathbf{q}}{\partial t}$ where $\frac{\partial \mathbf{q}}{\partial t}$ are the current joint velocities. We hope that the approach of both capturing and reproducing motion using the Lie group representation will allow robots to really move in a human like fashion leading to better human robot collaboration.

To further enhance the capabilities of the proposed approach, we can combine multiple FAMGP models as a way to prioritize over tasks. For example task and joint space models can be combined to produce a desired task space end-effector trajectory while moving the joints in

a human like manner. Since the predicted covariance increases unboundedly outside the range of the training data, chaining FAMGP models one after another may allow the robot to execute multiple intricate movements with smooth transitions such as hand writing. Furthermore, a multi-input FAMGP model will allow for additional inputs from a higher level controller to modify the resulting motion as it is executed. This can lead to a robot that can adjust its movements to express its priorities, intent, or emotions.

References

[1] J. Lin, D. Kulić, Human Pose Recovery using Wireless Inertial Measurement Units, Physiol Meas 33 (2012) 2099–2115.

[2] D. Kulić, G. Venture, K. Yamane, E. Demircan, I. Mizuuchi, K. Mombaur, Anthropomorphic movement analysis and synthesis: A survey of methods and applications, IEEE Transactions on Robotics 32 (2016) 776–795.

[3] A. Aristidou, J. Cameron, J. Lasenby, Real-time estimation of missing markers in human motion capture, in: 2008 2nd International Conference on Bioinformatics and Biomedical Engineering, IEEE, 2008, pp. 1343–1346.

[4] J. Steinbring, C. Mandery, N. Vahrenkamp, T. Asfour, U. D. Hanebeck, High-accuracy real-time whole-body human motion tracking based on constrained nonlinear kalman filtering, arXiv preprint arXiv:1511.04278 (2015).

[5] A. Aristidou, J. Lasenby, Real-time marker prediction and CoR estimation in optical motion capture, The Visual Computer 29 (2013) 7–26.

[6] V. Bonnet, G. Daune, V. Joukov, R. Dumas, P. Fraisse, D. Kulić, A. Seilles, S. Andary, G. Venture, A constrained extended Kalman filter for dynamically consistent inverse kinematics and inertial parameters identification, in: International Conference on Biomedical Robotics and Biomechatronics (BioRob), IEEE, 2016, pp. 952–957.

[7] S. Futamure, V. Bonnet, R. Dumas, D. Kulic, G. Venture, Dynamically consistent inverse kinematics framework using optimizations for human motion analysis, in: Humanoid Robots (Humanoids), 2016 IEEE-RAS 16th International Conference on, IEEE, 2016, pp. 436–441.

[8] H. A. Hashim, Special orthogonal group so (3), euler angles, angle-axis, rodriguez vector and unit-quaternion: Overview, mapping and challenges, arXiv preprint arXiv:1909.06669 (2019).

[9] L. W. Sy, M. Raitor, M. Del Rosario, H. Khamis, L. Kark, N. H. Lovell, S. Redmond, Estimating lower limb kinematics using a reduced wearable sensor count, IEEE Transactions on Biomedical Engineering (2020).

[10] S. Schaal, Is imitation learning the route to humanoid robots?, Trends in cognitive sciences 3 (1999) 233–242.

[11] B. D. Argall, S. Chernova, M. Veloso, B. Browning, A survey of robot learning from demonstration, Robotics and autonomous systems 57 (2009) 469–483.

[12] D. Clever, R. M. Schemschat, M. L. Felis, K. Mombaur, Inverse optimal control based identification of optimality criteria in whole-body human walking on level ground, in: Biomedical Robotics and

Biomechatronics (BioRob), 2016 6th IEEE International Conference on, IEEE, 2016, pp. 1192–1199.
[13] A.-S. Puydupin-Jamin, M. Johnson, T. Bretl, A convex approach to inverse optimal control and its application to modeling human locomotion, in: Robotics and Automation (ICRA), 2012 IEEE International Conference on, IEEE, 2012, pp. 531–536.
[14] J. Ćesić, V. Joukov, I. Petrović, D. Kulić, Full body human motion estimation on Lie groups using 3D marker position measurements, in: International conference on Humanoid Robots (Humanoids), IEEE-RAS, 2016, pp. 826–833.
[15] H. Zhou, H. Hu, Human motion tracking for rehabilitation—a survey, Biomedical Signal Processing and Control 3 (2008) 1–18.
[16] T. Sugihara, Solvability-unconcerned inverse kinematics by the Levenberg–Marquardt method, IEEE Transactions on Robotics 27 (2011) 984–991.
[17] J. Meyer, M. Kuderer, J. Müller, W. Burgard, Online marker labeling for fully automatic skeleton tracking in optical motion capture, in: Robotics and Automation (ICRA), 2014 IEEE International Conference on, IEEE, 2014, pp. 5652–5657.
[18] K. Ayusawa, Y. Ikegami, Y. Nakamura, Simultaneous global inverse kinematics and geometric parameter identification of human skeletal model from motion capture data, Mechanism and Machine Theory 74 (2014) 274–284.
[19] D. Kulić, G. Venture, K. Yamane, E. Demircan, I. Mizuuchi, K. Mombaur, Anthropomorphic Movement Analysis and Synthesis: A Survey of Methods and Applications, IEEE Trans Robot 32 (2016) 776–795.
[20] A. Filippeschi, N. Schmitz, M. Miezal, G. Bleser, E. Ruffaldi, D. Stricker, Survey of Motion Tracking Methods Based on Inertial Sensors: A Focus on Upper Limb Human Motion, Sensors 17 (2017) 1257.
[21] J. Bergmann, R. Mayagoitia, I. Smith, A portable system for collecting anatomical joint angles during stair ascent: a comparison with an optical tracking device., Dyn Med 8 (2009) 1–7.
[22] T. Seel, J. Raisch, T. Schauer, IMU-based joint angle measurement for gait analysis, Sensors 14 (2014) 6891–6909.
[23] R. Williamson, B. Andrews, Detecting absolute human knee angle and angular velocity using accelerometers and rate gyroscopes, Med Biol Eng Comput 39 (2001) 294–302.
[24] M. Boonstra, R. van der Slikke, N. Keijsers, R. van Lummel, M. de Waal Malefijt, N. Verdonschot, The Accuracy of Measuring the Kinematics of Rising from a Chair with Accelerometers and Gyroscopes, J Biomech 39 (2006) 354–358.
[25] H. Luinge, P. Veltink, Measuring Orientation of Human Body Segments using Miniature Gyroscopes and Accelerometers, Med Biol Eng Comput 43 (2005) 273–282.
[26] D. Roetenberg, H. Luinge, P. Slycke, Xsens mvn: Full 6dof human motion tracking using miniature inertial sensors, 2009.
[27] Y. Tian, X. Meng, D. Tao, D. Liu, C. Feng, Upper limb motion tracking with the integration of IMU and Kinect, Neurocomputing 159 (2015) 207–218.
[28] J. Bersamira, R. De Chavez, D. Salgado, M. Sumilang, E. Valles, E. Roxas, A. dela Cruz, Human Gait Kinematic Estimation based on Joint Data Acquisition and Analysis from IMU and Depth-

Sensing Camera, in: IEEE Int Conf Human Nano Info Tech Comm Contr Env Man, IEEE, 2019, pp. 1–6.
[29] C. Malleson, J. Collomosse, A. Hilton, Real-Time Multi-person Motion Capture from Multi-view Video and IMUs, Int J Comp Vis (2019).
[30] Ö. Bebek, M. Suster, S. Rajgopal, M. Fu, X. Huang, M. Cavusoglu, D. Young, M. Mehregany, A. van den Bogert, C. Mastrangelo, Personal Navigation via High-Resolution Gait-Corrected Inertial Measurement Units, IEEE Trans Instrum Meas 59 (2010) 3018–3027.
[31] Q. Yuan, I. Chen, 3-D Localization of Human Based on an Inertial Capture System, IEEE Trans Robot 29 (2013) 806–812.
[32] M. Dela Cruz, K. Legaspi, R. Marcelino, J. Rosete, D. Sangalang, C. Suarez, E. Roxas, K. Serrano, A. dela Cruz, Joint Gait Kinematic and Kinetic Analysis using Inertial Measurement Units and Plantar Pressure Sensor System, in: IEEE Int Conf Human Nano Info Tech Comm Contr Env Man, 2019, pp. 1–6.
[33] L. S. Vargas-Valencia, F. B. A. Schneider, A. G. Leal-Junior, P. Caicedo-Rodriguez, W. A. Sierra-Arevalo, L. E. Rodriguez-Cheu, T. Bastos-Filho, A. Frizera-Neto, Sleeve for Knee Angle Monitoring: An IMU-POF Sensor Fusion System, IEEE J Biomed Health In Print (2020).
[34] W. Kong, S. Sessa, S. Cosentino, M. Zecca, K. Saito, C. Wang, U. Imtiaz, Z. Lin, L. Bartolomeo, H. Ishii, T. Ikai, A. Takanishi, Development of a real-time IMU-based motion capture system for gait rehabilitation, in: IEEE Int Conf Robot Biomimetics, 2013, pp. 2100–2105.
[35] M. Benocci, L. Rocchi, E. Farella, L. Chiari, L. Benini, A wireless system for gait and posture analysis based on pressure insoles and Inertial Measurement Units, in: ICST Int Conf Perv Comput Tech Health, 2009.
[36] D. Novak, P. Reberšek, S. M. M. De Rossi, M. Donati, J. Podobnik, T. Beravs, T. Lenzi, N. Vitiello, M. C. Carrozza, M. Munih, Automated detection of gait initiation and termination using wearable sensors, Med Eng Phys 35 (2013) 1713–1720.
[37] J. Bae, M. Tomizuka, A tele-monitoring system for gait rehabilitation with an inertial measurement unit and a shoe-type ground reaction force sensor, Mechatronics 23 (2013) 646–651.
[38] V. Joukov, V. Bonnet, M. Karg, G. Venture, D. Kulić, Rhythmic EKF for Pose Estimation During Gait, in: IEEE-RAS Int Conf Hum Robot, 2015, pp. 1167–1172.
[39] V. Joukov, M. Karg, D. Kulic, Online tracking of the lower body joint angles using imus for gait rehabilitation, in: IEEE Int Conf Eng Med Biol Soc, 2014, pp. 2310–2313.
[40] S. Sprager, M. Juric, Inertial Sensor-Based Gait Recognition: A Review, Sensors 15 (2015) 22089–22127.
[41] M. Brandes, W. Zijlstra, S. Heikens, R. van Lummel, D. Rosenbaum, Accelerometry based assessment of gait parameters in children, Gait Posture 24 (2006) 482–486.
[42] R. González, A. López, J. Rodriguez-Uría, D. Álvarez, J. Alvarez, Real-time gait event detection for normal subjects from lower trunk accelerations, Gait Posture 31 (2010) 322–325.
[43] H.-C. Chang, Y.-L. Hsu, S.-C. Yang, J.-C. Lin, Z.-H. Wu, A Wearable Inertial Measurement System With Complementary Filter for Gait Analysis of Patients With Stroke or Parkinson's Disease, IEEE Access 4 (2016) 8442–8453.

[44] Q. Li, M. Young, V. Naing, J. M. Donelan, Walking speed and slope estimation using shank-mounted inertial measurement units, in: IEEE Int Conf Rehab Robot, 2009, pp. 839–844.
[45] P. Esser, H. Dawes, J. Collett, M. G. Feltham, K. Howells, Assessment of spatio-temporal gait parameters using inertial measurement units in neurological populations, Gait Posture 34 (2011) 558–560.
[46] F. Cavallo, A. Sabatini, V. Genovese, A step toward GPS/INS personal navigation systems: real-time assessment of gait by foot inertial sensing, in: IEEE/RSJ Int Conf Int Robot Syst, 2005, pp. 1187–1191.
[47] J. Rebula, L. Ojeda, P. Adamczyk, A. Kuo, Measurement of foot placement and its variability with inertial sensors, Gait Posture 38 (2013) 974–980.
[48] A. Kose, A. Cereatti, U. Della Croce, Bilateral step length estimation using a single inertial measurement unit attached to the pelvis, J NeuroEng Rehab 9 (2012) 9.
[49] D. Trojaniello, A. Cereatti, U. Della Croce, Accuracy, sensitivity and robustness of five different methods for the estimation of gait temporal parameters using a single inertial sensor mounted on the lower trunk, Gait Posture 40 (2014) 487–492.
[50] R. E. Mayagoitia, A. V. Nene, P. H. Veltink, Accelerometer and rate gyroscope measurement of kinematics: an inexpensive alternative to optical motion analysis systems, Biomech 35 (2002) 537–542.
[51] H. Rouhani, J. Favre, X. Crevoisier, K. Aminian, Measurement of Multi-segment Foot Joint Angles During Gait Using a Wearable System, J of Biomech Eng 134 (2012).
[52] S. Bakhshi, M. H. Mahoor, B. S. Davidson, Development of a body joint angle measurement system using IMU sensors, in: IEEE Int Conf Eng Med Biol Soc, IEEE, 2011, pp. 6923–6926.
[53] J. Castañeda, A. Ruiz-Olaya, C. Lara-Herrera, F. Roldán, Knee Joint Angle Monitoring System Based on Inertial Measurement Units for Human Gait Analysis, in: Lat Am Cong Biomed Eng, 2017, pp. 690–693.
[54] C. Bennett, C. Odom, M. Ben-Asher, Knee Angle Estimation based on IMU data and Artificial Neural Networks, in: Southern Biomedical Engineering Conference, 2013, pp. 111–112.
[55] V. Joukov, V. Bonnet, M. Karg, G. Venture, D. Kulic, Rhythmic Extended Kalman Filter for Gait Rehabilitation Motion Estimation and Segmentation, IEEE Trans Neural Syst Rehab Eng 26 (2018) 407–418.
[56] M. Miezal, B. Taetz, G. Bleser, Real-time inertial lower body kinematics and ground contact estimation at anatomical foot points for agile human locomotion, in: IEEE Int Conf Robot Autom, 2017, pp. 3256–3263.
[57] A. Szczesna, P. Pruszowski, Model-based extended quaternion Kalman filter to inertial orientation tracking of arbitrary kinematic chains, SpringerPlus 5 (2016) 1965.
[58] T. D. Barfoot, P. T. Furgale, Associating uncertainty with three-dimensional poses for use in estimation problems, IEEE Transactions on Robotics 30 (2014) 679–693.
[59] M. Benallegue, F. Lamiraux, Estimation and stabilization of humanoid flexibility deformation using only inertial measurement units and contact information, International Journal of Humanoid Robotics 12 (2015) 1550025.

[60] T. Lee, M. Leok, N. H. McClamroch, Global symplectic uncertainty propagation on SO(3), in: Conference on Decision and Control (CDC), 3, IEEE, 2008, pp. 61–66.
[61] A. W. Long, K. C. Wolfe, M. J. Mashner, G. S. Chirikjian, The Banana Distribution is Gaussian: A Localization Study with Exponential Coordinates, in: Robotics: Science and Systems (RSS), Sydney, Australia, 2012.
[62] G. Bourmaud, R. Mégret, M. Arnaudon, A. Giremus, Continuous-discrete extended Kalman filter on matrix Lie groups using concentrated Gaussian distributions, Journal of Mathematical Imaging and Vision 51 (2015) 209–228.
[63] C. Hertzberg, R. Wagner, U. Frese, L. Schröder, Integrating generic sensor fusion algorithms with sound state representations through encapsulation of manifolds, Information Fusion 14 (2013) 57–77.
[64] Q. Rentmeesters, P. A. Absil, P. Van Dooren, K. Gallivan, A. Srivastava, An efficient particle filtering technique on the Grassmann manifold, in: IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2010, pp. 3838–3841.
[65] A. M. Sabatini, Quaternion-based extended kalman filter for determining orientation by inertial and magnetic sensing, IEEE transactions on Biomedical Engineering 53 (2006) 1346–1356.
[66] R. A. Srivatsan, G. T. Rosen, D. F. N. Mohamed, H. Choset, Estimating se (3) elements using a dual quaternion based linear kalman filter., in: Robotics: Science and systems, 2016.
[67] Y. M. Lui, Human gesture recognition on product manifolds, The Journal of Machine Learning Research 13 (2012) 3297–3321.
[68] J. Martinez-Del-Rincon, M. Lewandowski, J. C. Nebel, D. Makris, Generalized Laplacian eigenmaps for modeling and tracking human motions, IEEE Transactions on Cybernetics 44 (2014) 1646–1660.
[69] M. Ding, G. Fan, Multilayer joint gait-pose manifolds for human gait motion modeling, IEEE Transactions on Cybernetics 45 (2015) 2413–2424.
[70] M. Devanne, H. Wannous, S. Berretti, P. Pala, M. Daoudi, A. Del Bimbo, 3-D human action recognition by shape analysis of motion trajectories on Riemannian manifold, IEEE Transactions on Cybernetics 45 (2015) 1340–1352.
[71] S. Brossette, A. Escande, G. Duchemin, B. Chrétien, A. Kheddar, Humanoid posture generation on non-euclidean manifolds, in: Humanoid Robots (Humanoids), 2015 IEEE-RAS 15th International Conference on, IEEE, 2015, pp. 352–358.
[72] D. M. Ste-Marie, B. Law, A. M. Rymal, O. Jenny, C. Hall, P. McCullagh, Observation interventions for motor skill learning and performance: an applied model for the use of observation, International Review of Sport and Exercise Psychology 5 (2012) 145–176.
[73] A. N. Meltzoff, M. K. Moore, Imitation of facial and manual gestures by human neonates, Science 198 (1977) 75–78.
[74] N. N. Wesch, B. Law, C. R. Hall, The use of observational learning by athletes, Journal of Sport Behavior 30 (2007) 219.
[75] J. Cumming, S. E. Clark, D. M. Ste-Marie, P. McCullagh, C. Hall, The functions of observational learning questionnaire (folq), Psychology of sport and exercise 6 (2005) 517–537.

[76] G. Di Pellegrino, L. Fadiga, L. Fogassi, V. Gallese, G. Rizzolatti, Understanding motor events: a neurophysiological study, Experimental brain research 91 (1992) 176–180.
[77] N. J. Hodges, I. M. Franks, Modelling coaching practice: the role of instruction and demonstration, Journal of sports sciences 20 (2002) 793–811.
[78] J. M. Mensch, C. D. Ennis, Pedagogic strategies perceived to enhance student learning in athletic training education, Journal of Athletic Training 37 (2002) S–199.
[79] J. Kober, J. Peters, Imitation and reinforcement learning, IEEE Robotics & Automation Magazine 17 (2010) 55–62.
[80] S. Schaal, A. Ijspeert, A. Billard, Computational approaches to motor learning by imitation, Philosophical Transactions of the Royal Society of London B: Biological Sciences 358 (2003) 537–547.
[81] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, S. Schaal, Dynamical movement primitives: learning attractor models for motor behaviors, Neural computation 25 (2013) 328–373.
[82] K. Muelling, J. Kober, J. Peters, Learning table tennis with a mixture of motor primitives, in: 2010 10th IEEE-RAS International Conference on Humanoid Robots, IEEE, 2010, pp. 411–416.
[83] J. Rosado, F. Silva, V. Santos, Adaptation of robot locomotion patterns with dynamic movement primitives, in: 2015 IEEE International Conference on Autonomous Robot Systems and Competitions, 2015, pp. 23–28.
[84] Z. Li, T. Zhao, F. Chen, Y. Hu, C.-Y. Su, T. Fukuda, Reinforcement learning of manipulation and grasping using dynamical movement primitives for a humanoidlike mobile manipulator, IEEE/ASME Transactions on Mechatronics 23 (2017) 121–131.
[85] M. Mistry, S. Schaal, Representation and control of the task space in humans and humanoid robots, in: Humanoid Robotics and Neuroscience: Science, Engineering and Society, CRC Press/Taylor & Francis, 2015.
[86] A. Paraschos, C. Daniel, J. R. Peters, G. Neumann, Probabilistic movement primitives, Advances in neural information processing systems 26 (2013) 2616–2624.
[87] S. Gomez-Gonzalez, G. Neumann, B. Schölkopf, J. Peters, Using probabilistic movement primitives for striking movements, in: 2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids), 2016, pp. 502–508.
[88] G. Maeda, M. Ewerton, R. Lioutikov, H. B. Amor, J. Peters, G. Neumann, Learning interaction for collaborative tasks with probabilistic movement primitives, in: 2014 IEEE-RAS International Conference on Humanoid Robots, IEEE, 2014, pp. 527–534.
[89] S. Calinon, A. Billard, Learning of gestures by imitation in a humanoid robot, Technical Report, Cambridge University Press, 2007.
[90] S. Calinon, A tutorial on task-parameterized movement learning and retrieval, Intelligent service robotics 9 (2016) 1–29.
[91] S. Calinon, F. D'halluin, E. Sauser, D. Caldwell, A. Billard, A probabilistic approach based on dynamical systems to learn and reproduce gestures by imitation, IEEE Robotics and Automation Magazine 17 (2010) 44–54.
[92] P. Englert, A. Paraschos, M. P. Deisenroth, J. Peters, Probabilistic model-based imitation learning, Adaptive Behavior 21 (2013) 388–403.

[93] E. Todorov, Optimality principles in sensorimotor control, Nature neuroscience 7 (2004) 907–915.
[94] N. Sylla, V. Bonnet, G. Venture, N. Armande, P. Fraisse, Human arm optimal motion analysis in industrial screwing task, in: Biomedical Robotics and Biomechatronics (2014 5th IEEE RAS & EMBS International Conference on, IEEE, 2014, pp. 964–969.
[95] M. L. Felis, K. Mombaur, Synthesis of full-body 3-d human gait using optimal control methods, in: Robotics and Automation (ICRA), 2016 IEEE International Conference on, IEEE, 2016, pp. 1560–1566.
[96] S.-H. Yeo, D. W. Franklin, D. M. Wolpert, When optimal feedback control is not enough: Feedforward strategies are required for optimal control with active sensing, PLoS computational biology 12 (2016) e1005190.
[97] R. S. Sutton, A. G. Barto, Reinforcement learning: An introduction, volume 1, MIT press Cambridge, 1998.
[98] P. Abbeel, A. Y. Ng, Apprenticeship learning via inverse reinforcement learning, in: Proceedings of the twenty-first international conference on Machine learning, ACM, 2004, p. 1.
[99] M. Wulfmeier, P. Ondruska, I. Posner, Deep inverse reinforcement learning, CoRR, abs/1507.04888 (2015).
[100] S. Levine, Z. Popovic, V. Koltun, Nonlinear inverse reinforcement learning with gaussian processes, Advances in neural information processing systems 24 (2011) 19–27.
[101] M. Alger, Deep Inverse Reinforcement Learning, Technical Report, The Australian National University, 2015.
[102] J. Stillwell, Naive Lie theory, Springer, 2008.
[103] J. M. Selig, Lie groups and Lie algebras in robotics, in: Computational Noncommutative Algebra and Applications, 2005, pp. 101–125.
[104] W. Park, Y. Wang, G. S. Chirikjian, The path-of-probability algorithm for steering and feedback control of flexible needles, The International Journal of Robotics Research 29 (2010) 813–830.
[105] G. Bourmaud, R. Mégret, A. Giremus, From intrinsic optimization to iterated extended Kalman filtering on Lie groups, Journal of Mathematical Imaging and Vision 55 (2016) 288–303.
[106] K. Tapp, Matrix Groups for Undergraduates, American Mathematical Society, 2005.
[107] Y. Wang, G. S. Chirikjian, Nonparametric second-order theory of error propagation on motion groups, The International Journal of Robotics Research 27 (2008) 1258–1273.
[108] G. S. Chirikjian, Stochastic Models, Information Theory, and Lie Groups, Volume 2: Analytic Methods and Modern Applications, Springer, 2012.
[109] G. Welch, G. Bishop, An Introduction to the Kalman Filter, Technical Report, Chapel Hill, NC, USA, 1995.
[110] S. J. Julier, The scaled unscented transformation, in: Proceedings of the 2002 American Control Conference (IEEE Cat. No. CH37301), volume 6, IEEE, 2002, pp. 4555–4559.
[111] C. E. Rasmussen, Gaussian processes in machine learning, in: Summer School on Machine Learning, Springer, 2003, pp. 63–71.
[112] C. Lantuéjoul, Geostatistical simulation: models and algorithms, Springer Science & Business Media, 2013.

[113] M. P. Deisenroth, D. Fox, C. E. Rasmussen, Gaussian processes for data-efficient learning in robotics and control, IEEE transactions on pattern analysis and machine intelligence 37 (2013) 408–423.
[114] D. Nguyen-Tuong, M. Seeger, J. Peters, Model learning with local gaussian process regression, Advanced Robotics 23 (2009) 2015–2034.
[115] J. Gonzalvez, E. Lezmi, T. Roncalli, J. Xu, Financial applications of gaussian processes and bayesian optimization, arXiv preprint arXiv:1903.04841 (2019).
[116] C. E. Rasmussen, Gaussian processes for machine learning (2006).
[117] J. Quiñonero-Candela, C. E. Rasmussen, A unifying view of sparse approximate gaussian process regression, Journal of Machine Learning Research 6 (2005) 1939–1959.
[118] A. J. Smola, P. L. Bartlett, Sparse greedy gaussian process regression, in: Advances in neural information processing systems, 2001, pp. 619–625.
[119] M. Seeger, C. Williams, N. Lawrence, Fast forward selection to speed up sparse Gaussian process regression, Technical Report, 2003.
[120] E. Snelson, Z. Ghahramani, Sparse gaussian processes using pseudo-inputs, in: Advances in neural information processing systems, 2006, pp. 1257–1264.
[121] M. Titsias, Variational learning of inducing variables in sparse gaussian processes, in: Artificial Intelligence and Statistics, 2009, pp. 567–574.
[122] C.-A. Cheng, B. Boots, Variational inference for gaussian process models with linear complexity, in: Advances in Neural Information Processing Systems, 2017, pp. 5184–5194.
[123] H. Salimbeni, C.-A. Cheng, B. Boots, M. Deisenroth, Orthogonally decoupled variational gaussian processes, in: Advances in neural information processing systems, 2018, pp. 8711–8720.
[124] J. Shi, M. Titsias, A. Mnih, Sparse orthogonal variational inference for gaussian processes, in: International Conference on Artificial Intelligence and Statistics, 2020, pp. 1932–1942.
[125] J. Hartikainen, S. Särkkä, Kalman filtering and smoothing solutions to temporal gaussian process regression models, in: 2010 IEEE international workshop on machine learning for signal processing, IEEE, 2010, pp. 379–384.
[126] A. Grigorievskiy, N. Lawrence, S. Särkkä, Parallelizable sparse inverse formulation gaussian processes (spingp), in: 2017 IEEE 27th International Workshop on Machine Learning for Signal Processing (MLSP), IEEE, 2017, pp. 1–6.
[127] V. Adam, S. Eleftheriadis, A. Artemev, N. Durrande, J. Hensman, Doubly sparse variational gaussian processes, in: International Conference on Artificial Intelligence and Statistics, 2020, pp. 2874–2884.
[128] M. Lázaro-Gredilla, J. Quiñonero-Candela, C. E. Rasmussen, A. R. Figueiras-Vidal, Sparse spectrum gaussian process regression, The Journal of Machine Learning Research 11 (2010) 1865–1881.
[129] C. Williams, M. Seeger, Using the nyström method to speed up kernel machines, Advances in neural information processing systems 13 (2000) 682–688.
[130] H. Peng, Y. Qi, Eigengp: Gaussian process models with adaptive eigenfunctions, arXiv preprint arXiv:1401.0362 (2014).

[131] V. Joukov, D. Kulic, Gaussian process based model predictive controller for imitation learning, in: 2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids), IEEE, 2017, pp. 850–855.
[132] S. Rezaei-Shoshtari, D. Meger, I. Sharf, Cascaded gaussian processes for data-efficient robot dynamics learning, arXiv preprint arXiv:1910.02291 (2019).
[133] G. Cao, E. M.-K. Lai, F. Alam, Gaussian process model predictive control of an unmanned quadrotor, Journal of Intelligent & Robotic Systems 88 (2017) 147–162.
[134] E. V. Bonilla, K. M. Chai, C. Williams, Multi-task gaussian process prediction, in: Advances in neural information processing systems, 2008, pp. 153–160.
[135] X. Wang, W. Yang, B. Sun, Derivatives of kronecker products themselves based on kronecker product and matrix calculus, Journal of Theoretical and Applied Information Technology 48 (2013).
[136] C. E. Garcia, D. M. Prett, M. Morari, Model predictive control: theory and practice—a survey, Automatica 25 (1989) 335–348.
[137] V. Joukov, J. Ćesić, K. Westermann, I. Marković, I. Petrović, D. Kulić, Estimation and observability analysis of human motion on lie groups, IEEE transactions on cybernetics 50 (2019) 1321–1332.
[138] Y. Bar-Shalom, T. Kirubarajan, X.-R. Li, Estimation with Applications to Tracking and Navigation, John Wiley & Sons, Inc., 2002.
[139] M. W. Spong, S. Hutchinson, M. Vidyasagar, Robot Modeling and Control, John Wiley & Sons, Inc., 2006.
[140] W. L. Brogan, Modern Control Theory (3rd Ed.), Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1991.
[141] P. S. Maybeck, Stochastic Models: Estimation and Control: Volume 1, number vol. 1 in Mathematics in Science and Engineering, Elsevier Science, 1979.
[142] W. J. Rugh, Linear System Theory (2Nd Ed.), Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1996.
[143] Y. Kawano, T. Ohtsuka, Nonlinear eigenvalue approach to differential Riccati equations for contraction analysis, IEEE Transactions on Automatic Control (accepted for publication) 62 (2017) 6497–6504.
[144] F. M. Mirzaei, S. I. Roumeliotis, A Kalman filter-based algorithm for IMU-camera calibration: Observability analysis and performance evaluation, IEEE Transactions on Robotics 24 (2008) 1143–1156.
[145] R. Hermann, A. Krener, Nonlinear controllability and observability, IEEE Transactions on Automatic Control 22 (1977) 728–740.
[146] P. Bonnifait, G. Garcia, Design and experimental validation of an odometric and goniometric localization system for outdoor robot vehicles, IEEE Transactions on Robotics and Automation 14 (1998) 541–548.
[147] A. Martinelli, R. Siegwart, Observability analysis for mobile robot localization, in: International Conference on Intelligent Robots and Systems (IROS), IEEE/RSJ, 2005, pp. 1471–1476.

[148] K. W. Lee, W. S. Wijesoma, J. Ibanez Guzman, On the observability and observability analysis of SLAM, in: International Conference on Intelligent Robots and Systems (IROS), IEEE/RSJ, 2006, pp. 3569–3574.
[149] A. Martinelli, D. Scaramuzza, R. Siegwart, Automatic self-calibration of a vision system during robot motion, in: International Conference on Intelligent Robots and Systems (IROS), IEEE/RSJ, 2006, pp. 43–48.
[150] J. Kelly, G. S. Sukhatme, Visual–inertial sensor fusion: Localization, mapping and sensor-to-sensor self-calibration, The International Journal of Robotics Research 30 (2011) 56–79.
[151] J. A. Hesch, D. G. Kottas, S. L. Bowman, S. I. Roumeliotis, Camera-IMU-based localization: observability analysis and consistency improvement, The International Journal of Robotics Research 33 (2013) 182–201.
[152] V. Joukov, R. D'Souza, D. Kulić, Human pose estimation from imperfect sensor data via the extended kalman filter, in: International Symposium on Experimental Robotics, Springer, 2016, pp. 789–798.
[153] S. J. Julier, J. K. Uhlmann, New extension of the kalman filter to nonlinear systems, in: Signal processing, sensor fusion, and target recognition VI, volume 3068, International Society for Optics and Photonics, 1997, pp. 182–194.
[154] D. Q. Huynh, Metrics for 3D rotations: Comparison and analysis, Journal of Mathematical Imaging and Vision 35 (2009) 155–164.
[155] F. De la Torre, J. Hodgins, A. Bargteil, X. Martin, J. Macey, A. Collado, P. Beltran, Guide to the Carnegie Mellon University multimodal activity (CMU-MMAC) database, 2008.
[156] V. Joukov, M. Karg, D. Kulić, Online tracking of the lower body joint angles using IMUs for gait rehabilitation, in: IEEE Engineering in Medicine and Biology Conference, 2014, pp. 2310–2313.
[157] D. Tedaldi, A. Pretto, E. Menegatti, A robust and easy to implement method for IMU calibration without external equipments, in: ICRA, IEEE, 2014, pp. 3042–3049.
[158] M. Rickert, A. Gaschler, Robotics Library: An object-oriented approach to robot applications, in: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, Canada, 2017, pp. 733–740.
[159] J. Carpentier, G. Saurel, G. Buondonno, J. Mirabel, F. Lamiraux, O. Stasse, N. Mansard, The pinocchio c++ library – a fast and flexible implementation of rigid body dynamics algorithms and their analytical derivatives, in: IEEE International Symposium on System Integrations (SII), 2019.
[160] D. E. Krebs, C. E. Robbins, L. Lavine, R. W. Mann, Hip biomechanics during gait, Journal of Orthopaedic & Sports Physical Therapy 28 (1998) 51–59.
[161] V. Joukov, J. Lin, D. Kulić, Closed-chain pose estimation from wearable sensors, in: IEEE-RAS Int Conf Human Rob, 2019.
[162] N. Gupta, R. Hauser, Kalman filtering with equality and inequality state constraints, 2007.
[163] V. Joukov, J. F.-S. Lin, K. Westermann, D. Kulić, Real-time unlabeled marker pose estimation via constrained extended Kalman filter, in: Int Symp Exp Robot, 2018, pp. 762–771.
[164] Y. Zheng, K. Chan, C. Wang, Pedalvatar: An IMU-based real-time body motion capture system using foot rooted kinematic model, in: IEEE/RSJ Int Conf Int Robot Syst, 2014, pp. 4130–4135.

[165] M. Harrington, A. Zavatsky, S. Lawson, Z. Yuan, T. Theologis, Prediction of the hip joint centre in adults, children, and patients with cerebral palsy based on magnetic resonance imaging, J Biomech 40 (2007) 595–602.
[166] D. Huynh, Metrics for 3D Rotations: Comparison and Analysis, J Math Imaging Vis 35 (2009) 155–164.
[167] V. Joukov, J. F.-S. Lin, D. Kulić, Generalized hebbian algorithm for wearable sensor rotation estimation, in: IEEE-RAS Int Conf Int Rob Syst, 2017, pp. 2248–2253.
[168] H. Q. Minh, P. Niyogi, Y. Yao, Mercer's theorem, feature maps, and smoothing, in: International Conference on Computational Learning Theory, Springer, 2006, pp. 154–168.
[169] A. McHutchon, Differentiating gaussian processes (2013).
[170] A. Niati, Inverse of sum of kronecker products as a sum of kronecker products, GPS Solutions 23 (2019) 2.
[171] G. E. Fasshauer, M. J. McCourt, Kernel-based approximation methods using Matlab, volume 19, World Scientific Publishing Company, 2015.
[172] G. E. Fasshauer, M. J. McCourt, Stable evaluation of gaussian radial basis function interpolants, SIAM Journal on Scientific Computing 34 (2012) A737–A762.
[173] A. J. Smola, B. Schölkopf, K.-R. Müller, The connection between regularization operators and support vector kernels, Neural networks 11 (1998) 637–649.
[174] H. Prodinger, Representing derivatives of chebyshev polynomials by chebyshev polynomials and related questions (2017).
[175] D. R. Burt, Spectral Methods in Gaussian Process Approximations., Master's thesis, University of Cambridge, 2018.
[176] S. M. Khansari-Zadeh, A. Billard, Learning stable nonlinear dynamical systems with gaussian mixture models, IEEE Transactions on Robotics 27 (2011) 943–957.
[177] F. Vivarelli, Studies on the generalisation of Gaussian processes and Bayesian neural networks, Ph.D. thesis, Aston University, 1998.
[178] A. Lederer, J. Umlauft, S. Hirche, Posterior variance analysis of gaussian processes with application to average learning curves, arXiv preprint arXiv:1906.01404 (2019).
[179] M. W. Spong, S. Hutchinson, M. Vidyasagar, et al., Robot modeling and control, 2006.
[180] A. Paraschos, C. Daniel, J. Peters, G. Neumann, Using probabilistic movement primitives in robotics, Autonomous Robots 42 (2018) 529–551.
[181] O. Dermy, A. Paraschos, M. Ewerton, J. Peters, F. Charpillet, S. Ivaldi, Prediction of intention during interaction with icub with probabilistic movement primitives, Frontiers in Robotics and AI 4 (2017) 45.
[182] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, S. Boyd, OSQP: An operator splitting solver for quadratic programs, Mathematical Programming Computation (2020).
[183] E. Eade, Lie groups for 2d and 3d transformations, URL http://ethaneade. com/lie. pdf, revised Dec 117 (2013) 118.
[184] D. Condurache, I.-A. Ciureanu, Closed form of the baker-campbell-hausdorff formula for the lie algebra of rigid body displacements, in: European Congress on Computational Methods in Applied Sciences and Engineering, Springer, 2019, pp. 307–314.