Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

b-it Bonn-Aachen International Center for Information Technology

R&D Project

Extending the Vereshchagin hybrid dynamic solver to mobile robots

Sushma Devaramani

Submitted to Hochschule Bonn-Rhein-Sieg,
Department of Computer Science
in partial fullfilment of the requirements for the degree
of Master of Science in Autonomous Systems

Supervised by

Prof. Dr. Paul G. Plöger
Sven Schneider

January 2019

I, the undersigned below, declare that this work has not previously been submitted to this or any other university and that it is, unless otherwise stated, entirely my own work.

Date	Sushma Devaramani

Abstract

The objective of the project is to extend and apply the Vereshchagin hybrid dynamic solver to mobile robots. A typical execution of mobile robot tasks involves navigation from one point to another by effectively avoiding obstacles. In autonomous systems, there are various algorithms employed to implement collision avoidance. These approaches follow velocity-based control scheme, which primarily aims at ignoring physical contact with the objects around the robot. However, if the situation demands physical contact robot must not cause any damage to the environment. However, when the robot comes across an obstacle unexpectedly, the velocity control strategy fails. The reason for failure is that the control scheme cannot instantly detect the object and control the robot motions. Therefore, there is a need to include safety constraints which the robot must handle while executing its functions. The issue of handling safety constraints has been addressed in robot manipulators for ages since they are continuously involved in manipulating the objects in the world. Additionally, the diversity of robot motion tasks has led to the development of (constrained) task control methodologies with origins in force control, humanoid robot control, mobile manipulator control, visual servoing, etc. The sequence of tasks such as pick and place operations in manipulators are executed through task specification strategy, where each of the associated task constraints is modeled. Nevertheless, there is no specific task specification approach employed in mobile robots. In robot manipulators, there are several software frameworks, algorithms and dynamic solvers employed to realize the task constraints instantly and efficiently. The Popov-Vereshchagin solver is one such dynamic solvers practiced by manipulators. The Vereshchagin is a significant algorithm for the posture control of mobile manipulators and humanoid robots since such tasks typically require specifications of motion and/or force constraints on the end-effectors and other segments. Additionally, the Vereshchagin solver can be applied to closed as well as open kinematic chains. Since the wheels and base of

the mobile robot can be modeled as a closed kinematic chain, the solver can be extended and applied to mobile robots.

Acknowledgements

Thanks to ....

Contents

List of Figures

List of Tables

Acronyms

SoT	-	Stack of Tasks
WBC	-	Whole Body Control
iTaSC	-	instantaneous Task Specification and Control
WBOSC	-	Whole Body Operational Space Control
OCP	-	Optimal Control Problem

List of symbols

M	Inertia matrix that maps between joint space domain and force domain
q	Joint position vector
$\dot{q}$	Joint velocity vector
$\ddot{q}$	Joint acceleration vector
f_c	Joint space constraint forces
$\hat{b}(q, \dot{q})$	Bias acceleration over second order derivative of holonomic position constraint
τ_a	Input forces
τ_c	Constraint forces
$C(q, \dot{q})$	Bias forces
$\ddot{X}$	Cartesian acceleration
H_i	Inertial matrix of link i
U_i	Vector comprising of Coriolis and centrifugal forces
d	Moment of rotor inertia
Q	Generalized forces
A_N	Linear constraint matrix of order 6 x m where m is the number of constraints on a segment
b_N	Acceleration energy (force times acceleration)

1

Introduction

Safety is one of the critical factors to be considered when designing robotic systems in human environments [9]. The robotic engineers and researchers from an extended period, have focused on the safety of robots and its workspace. The growing application of the two main classes of robots, i.e., manipulators and mobile robots in diverse fields adds to the necessity for safety.

Robot manipulators are widely employed in an industrial environment. As manipulators are bulky and dangerous, the tasks are confined to a closed environment, away from humans. However, recently, the advancement in the field of manipulators has contributed to a safe interaction with humans. The increasing complexity in tasks has led to never-ending research in the collision handling systems. For instance, a robotic arm performing pick and place operation in a structured environment has to plan and execute the task safely by achieving dynamic collision avoidance. Additionally, there are various constraints imposed by the task specification. One such constraint would be to place the object vertically on the table without damaging the object and the workspace. Likewise, many such constraints are imposed as the complexity of tasks increases. There are software frameworks and dynamic solvers targeted to realize the constraints in real-time.

Consequently, in the field of autonomous mobile robots, safe navigation is the crucial goal [6]. Due to their ability to navigate, mobile robots are often employed in applications such as logistics, security and defense, inspection and maintenance, cleaning, agriculture and many more. Typically navigating in populated environments, the mobile robot performs a task under changing external circumstances. Therefore, the robots must plan dynamically to respond to such unforeseen situations [6].

YET TO WRITE..............

1.1 Motivation

The robot navigation has been implemented effectively by many approaches. However, these methods often perform obstacle avoidance [19] [8]. And the robot motions are controlled at velocity-level. In some circumstances, the objectives demand force/acceleration constraints if the robot is obliged to come in contact with the environment. The traditional velocity-based control cannot handle the constraints in force/acceleration level. Hence there is a necessity to manage these constraints in mobile robots. In contrast to mobile robots, the need for continuous physical interaction with the environment has already been recognized for several decades in the manipulators. This field is well researched in robotic arms that manipulate objects. The arm/joint parameters are bound by specific force constraints [9]. Specifically, the end-effector joints are limited by allowable force on the object. For instance, consider a pick and place scenario, where the arm has to grasp a fragile glass and place it on a workbench. Here, the end-effector has to grip with a specific force such that the glass neither breaks nor slips out. Additionally, the task might impose multiple constraints, such as the end-effector must place the object perpendicular to the plane by applying limited forces. The arm must satisfy these dynamic constraints. The controller supervises these constraints at that instant of time. Besides, many such task constraints are imposed and hence dynamic solvers are used to realize them instantly.

The Vereshchagin solver is one such dynamic solver that can handle the requirements presented above in robotic manipulators. The aim of the project is to extend and apply this solver to mobile robots.

1.2 Challenges and Difficulties

YET TO BE WRITTEN!!

1.3 Problem Statement

The robot manipulators are extensively involved in physical interactions with the objects in the environment. There are various software frameworks and dynamic solvers to manage the task constraints while performing manipulation tasks. However, the mobile robots do not exhibit direct physical contact with the world unless in two circumstances,

- When the robot comes across an obstacle unexpectedly;
  Use case: Consider an autonomous system navigating from point A to point B by avoiding obstacles. When the robot has to turn around a corner, it is unaware of any approaching obstacles. In such situations, the base must exhibit motions with limited force. Even if the robot comes in contact with the obstacle, it should not harm the environment.
- When the robot task involves contact with an object;
  1. **Use case:** Consider a multi-robot system performing logistic tasks in an industrial environment, where a robot has to join itself to another through some means (e.g., a hook). For this purpose, the robot initially has to align and come in contact with the other robot physically to connect itself. In this example, the task demands constraints such as safe alignment with limited acceleration.
  2. **Use case:** Consider a wall alignment problem. The usual procedure is to detect the wall, and the mounted sensors continuously compute the distance values from the wall. Based on these values, the robot adjusts its position. In spite of this traditional method, the project presents an approach to exploit the obstacle. If a virtual force is pushing the robot towards the wall, and at one point it comes in contact with the wall. There is an acceleration constraint when it tries to move further. The solver equipped in the article utilizes this constraint to align the robot to the wall.

The project addresses the safety constraints in the situations as explained in use cases. The project also addresses the issue regarding task specification for mobile bases. Generally, the manipulators involve a task specification strategy to fulfill the sequence of tasks. These tasks impose several constraints on the robot actions. Many software frameworks handle these constraints at the task level. However, in the field of mobile robots, there is no practical implementation of task specification approach. Below is a use case that depicts why task specification procedure would be helpful for mobile bases.

- **Use case:** A mobile robot is performing logistic functions in a hospital environment. The task requirement is to carry objects to a destination. Limited velocity and forces constrain the robot motions. Additionally, the robot must drive inside a specified boundary. The robot must effectively be able to handle them instantly. The project presents a similar approach to task specification for mobile bases.
  The project seeks to solve the issue of handling the constraints arising from multiple tasks.

2

State of the Art

The State of the art section discusses the various software frameworks and dynamic solvers employed in robot manipulators to dynamically realize constraints originating from task specification.

The kinematic and dynamic solvers compute control inputs for a manipulator given the manipulator's dynamic parameters such as link inertia or joint friction, and external forces acting on the manipulator. They realize the task constraints that originate from task specification. Similarly, a mobile base naturally resembles a partially-constrained floating base. The constrained degrees of freedom of motions are,

- linear acceleration perpendicular to the ground;
- angular acceleration about the two axes that are parallel to the ground.

Additionally, each wheel introduces a *Cartesian acceleration constraint* known as the *sliding constraint*. In order to realize these motion constraints in mobile robots, the paper presents an extension to **Popov-Vereshchagin** hybrid dynamic solver. The solver computes the stabilized control inputs in manipulators, given the task objectives, the motion model of the robot as well as the cost function. The previous methods to address the navigation problem such as path tracking, point-point stabilization etc were controlled at kinematic level. There are few current approaches that explicitly control the robot at a dynamic level. One of the papers addresses the general problem of path tracking in mobile robots by presenting a detailed dynamic model with torque coupling. But the article fails to address the complex inner loop dynamics.

2.1 Rigid Body Dynamics

....

2.2 Software Frameworks

2.2.1 Task Frame Formalism (TFF)

TFF is an intuitive programming framework for manipulator applications. It is part of the robot control architecture to transform the high-level commands to unambiguous low-level control parameters current approaches aim at executing complex robot tasks by specifying various compliant motion commands for manipulators. These high-level commands have to be interpreted as atomic robot actions, i.e. no further subdivision is possible. The implementations are primarily focused in the field of the robot manipulator. Manipulation primitives can be linked together to a sequence or complex nets, which describe a robot task. By the execution of these primitives, a huge variety of very complex robot tasks can be covered [3]-[6].
Within this framework, the whole space of possible task directions can be divided into two orthogonal subspaces: one composed of force-controlled directions (position-constrained), and the other that represents velocity-controlled directions (force-constrained). A great number of tasks can be performed within this framework. The only condition is that it must be possible to decompose the task into force and velocity- controlled directions.

The drawback of the task frame approach is that it only applies to task geometries for which separate control modes can be assigned independently to three pure translational and three pure rotational directions along the axes of a single frame. A more systematic approach is to assign control modes and corresponding constraints to arbitrary directions in the six-dimensional manipulation space.

2.2.2 Operational Space Formulation

The operational space of a manipulator is defined by the configuration space of the end-effector (the standard cartesian space). Operational space control (OSC) is an approach to manipulator control that focuses on the dynamic behavior of a serial rigid-link manipulator as seen at the end effector as it evolves in its operational space [7].

The description, analysis, and control of manipulator systems with respect to the dynamic characteristics of their end-effectors have been the basic motivation in the research and development of the operational space formulation framework [8].

An operational coordinate system is a set of x of m independent parameters describing the manipulator end-effector position and orientation in the frame R0. The end-effector equations of motion in operational space are [9],

$$\lambda(q)\ddot{x} + \mu(q, \dot{q}) + p(q) = F_{op}$$

Where, $\lambda(q)$ is the kinetic energy matrix of the system with respect to the operational point. x defines the operational space coordinates of the end-effector. $\mu(q, \dot{q})$ represents the Coriolis and centrifugal forces acting at the same operational point. $p(q)$ depicts the gravitational forces also expressed at that point. F_{op} is the generalized force vector expressed in the operational space.

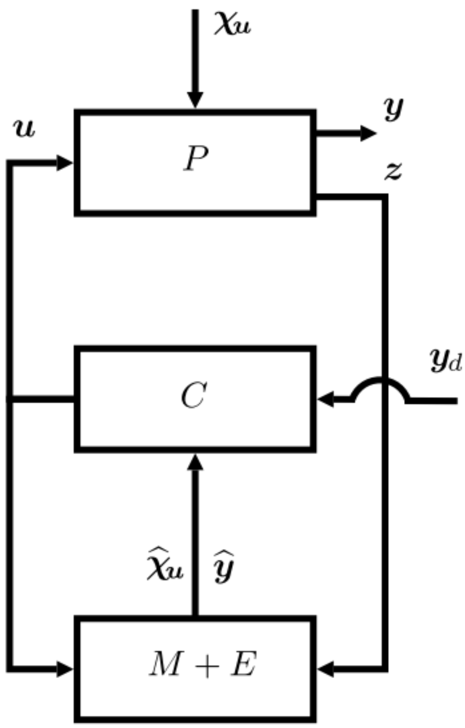
One drawback of the approach is that it directly controls the manipulator in its functional / task space rather than controlling in corresponding joint space that occurs only after geometric and kinematic transformations [10].

2.2.3 Stack of Tasks(SoT)

.....

2.2.4 Instantaneous task specification using constraints (iTaSC)

(De Schutter et al.) was the first to introduce a systematic constraint-based procedure to specify complex tasks for a variety of sensor-based robotic systems [1].



The iTaSC is a is a systematic constraint-based approach to specify complex tasks of general sensor-based robot systems. iTASC integrates both instantaneous task specification and estimation of geometric uncertainty in a unified framework. It allows easy specification and code-generation for robot tasks. It generates robot motions by specifying constraints between (parts of) the robots and their environment. iTaSC was born as a specification formalisms to generalize and extend existing approaches, such as the Operational Space Approach, the Task Function Approach, the Task Frame Formalism, geometric Cartesian Space control, and Joint Space control.

The key advantages of iTaSC over traditional motion specification methodologies are [11]:

- *Composability of constraints:* Multiple constraints can be combined, hence the constraints can be partial, i.e. they do not have to constrain the full 6D relation between two objects.
- *Re-usability of constraint specification:* The constraints specify a relation between feature frames, which have a semantic meaning in the context of a task, implying that the same task specification can be reused on different objects.
- *Automatic derivation of the control solution:* The iTaSC methodology generates a robot motion that optimizes the constraints by automatically deriving the controllers from the constraint specification.

The iTaSC concepts apply to specifications in the robot, Cartesian and sensor space, to the position, velocity or torque-controlled robots, to explicit and implicit specifications, and to equality and inequality constraints. The current implementation, however, is currently still limited to the velocity control and equality constraints subset.

2.3 Software architecture

2.3.1 Whole body control

....

2.4 Limitations of previous work

Software frameworks: The primitive software frameworks implemented in the area of robot manipulators to handle the constraints originating from task specification are,

- Task frame formalism(TFF)
- The Operational Space Formulation

- Instantaneous task specification using constraints(iTaSC)
- Stack Of Tasks(SOT)

3

Popov-Vereshchagin Hybrid Dynamics Solver

3.1 Solver Interpretation

Popov et al. [10] first introduced the solver in 1989. This linear-time constrained hybrid-dynamic solver is based on one of the principles of mechanics - *Gauss principle of least constraints* [10], that formulates a "dynamically natural way" to solve the redundancy problem in manipulators [4]. At the basis, the principle states that "*The constrained motion of a system or mechanism is the closest possible acceleration to free motion (unconstrained) that corresponds to the unique minimum of a convex function.*" [7]. Here, the solver computes the true acceleration of the constrained kinematic chain by minimizing the *acceleration energy*.

As defined by the task requirements in a manipulator, various cartesian acceleration constraints are imposed on one or more segments. Physically, these constraints are forces exerted to limit the motion of segments in certain direction, which in turn produces acceleration energy. The solver aims to realize these constraints by minimizing the acceleration energy using domain-specific computational sweeps [8].

The solver computes the solution to a constrained system that can be formulated as [8],

$$M(q)\ddot{q} + f_c = \tau_a(q) - C(q, \dot{q}) \tag{3.1}$$

The equation 3.1 is derived from the robot's dynamic motion model [8]. See the appendix section A for the complete explanation. Here $M(q)$ represents inertial matrix that maps from joint space ($\ddot{q}$) to force space (τ). The term f_c denotes constraint forces acting on the joints.

As previously mentioned, the solver minimizes the Gauss function to resolve the redundancy problem in manipulators. It is given by [10],

$$Z = \min_{\ddot{q}} \Big\{ \sum_{i=0}^{N} \frac{1}{2} \ddot{X}_i{}^T H_i \ddot{X}_i + U_i^T \ddot{X}_i + \sum_{i=1}^{N} \frac{1}{2} d_i \ddot{q}_i{}^2 - Q_i \ddot{q}_i \Big\} \tag{3.2}$$

This Gauss function (Z) is subjected to linear constraints given by [8],

$$A_N^T \ddot{X}_N = b_N \tag{3.3}$$

In the equation 3.2, Z is the acceleration energy of kinematic chain, also called as *Zwang* [8]. The function Z is minimized with respect to joint accelerations, $\ddot{q}$. $\ddot{X}_i$ represents 6 x 1 Cartesian acceleration vector of segment i, expressed in Plücker coordinates (See the appendix section B). H_i is the cartesian space inertial matrix. U_i is a vector of bias forces (i.e., external forces, Coriolis and centrifugal forces) acting on segment i.

In the equation 3.3, A_N is a matrix of order 6 x m with m as the number of constraints. The columns of the matrix represents the direction of constraint forces experienced by the end-effector. b_N is a vector of order m x 1 and is called acceleration energy set-point. The units of b_N is *acceleration times force.*

The Vereshchagin solver is domain-specific, since it solves the constrained optimization problem in the domain of rigid body dynamics. Evaluating for a possible minimum solution to the function 3.2, in presence of certain linear constraints is termed as *constrained optimization problem.* The equation is thus extended to [8],

$$Z = \min_{\ddot{q}} \Big\{ \sum_{i=0}^{N} \frac{1}{2} \ddot{X}_i{}^T H_i \ddot{X}_i + U_i^T \ddot{X}_i + \sum_{i=1}^{N} \frac{1}{2} d_i \ddot{q}_i{}^2 - Q_i \ddot{q}_i + \nu_T A_N^T \ddot{X}_N \Big\} \tag{3.4}$$

Z is a quadratic function and is minimized by applying the method of Lagrange multipliers [2], where ν is the non-negative *Lagrange multiplier*. In further steps, the solver is derived based on the *Bellman's principle of optimality* [3] [1]. The equation is reformulated as [8],

$$Z_{i-1}(\ddot{X}_{i-1}, \nu) = \min_{\ddot{q}} \left\{ \frac{1}{2}\ddot{X}_{i-1}^{T} H_{i-1} \ddot{X}_{i-1} + U_i^T \ddot{X}_i + \frac{1}{2} d_i \ddot{q}_i{}^2 - Q_i \ddot{q}_i + Z_i(\ddot{X}_i, \nu) \right\} \quad (3.5)$$

On further solving the equation 3.5 and minimizing with respect to $\ddot{q}$ will yield the solution to a constrained dynamics problem, which is of the form [8],

$$F_c = A_N \nu \quad (3.6)$$

where, F_c is the vector of constraint forces imposed on the segments.

To determine the true acceleration of the constrained kinematic chain at every instance of time, the Vereshchagin solver applies computational sweeps. Through these outward and inward recursions, the solver visits every segments(links) and returns *joint accelerations* ($\ddot{q}$), *Cartesian accelerations* ($\ddot{X}$) and joint torques ($\tau_{control}$) as the solution to the constrained dynamics problem [8].

3.2 Algorithm Description

The algorithm illustrating the computational sweeps in the Vereshchagin solver is described in this section. As specified, the algorithm comprises three recursions - outward, inward and outward. Here, the outward recursion refers to traversing from the fixed base of a kinematic chain to its end-effector. Contrarily, the inward recursion loops from end-effector to base. Through these recursions, the solver computes the dynamics of a kinematic chain that is subjected to acceleration constraints.

The complete algorithm is given below [8] [10] [11],

Algorithm 1: Constrained Hybrid Dynamic Solver

Input : Robot geometry, inertial data, q_i, $\dot{q}_i$, τ_i, $\ddot{X}_0$, F_i^{ext}, A_N, b_N

Output : $\tau_{control}$, $\ddot{q}_i$, $\ddot{X}_i$

begin

/* *Outward sweep of pose, twist and bias components* */

for $i \leftarrow 0$ *to* $N-1$ **do**

$\quad {}^{i+1}_{i}X = \left({}^{d_i}_{i}X \, {}^{i+1}_{d_i}X(q_i)\right);$

$\quad \dot{X}_{i+1} = {}^{i+1}X_i \dot{X}_i + S_{i+1}\dot{q}_{i+1};$

$\quad \ddot{X}_{bias,i+1} = \dot{X}_{i+1} \times S_{i+1}\dot{q}_{i+1};$

$\quad F^b_{bias,i+1} = \dot{X}_{i+1} \times^* H_{i+1}\dot{X}_{i+1} - {}^{i+1}X^*_0 F^{ext}_0;$

$\quad H^A_{i+1} = H_{i+1};$

$\quad F^A_{bias,i+1} = F^b_{bias,i+1};$

end

/* *Inward sweep of inertia and force* */

for $i \leftarrow (N-1)$ *to* 0 **do**

$\quad D_{i+1} = d_{i+1} + S^T_{i+1} H^A_{i+1} S_{i+1};$

$\quad P^A_{i+1} = 1 - H^A_{i+1} S_{i+1} D^{-1}_{i+1} S^T_{i+1};$

$\quad H^a_{i+1} = P^A_{i+1} H^A_{i+1};$

$\quad H^A_i = H^A_i + \sum {}^iX^T_{i+1} H^a_{i+1} \, {}^iX_{i+1};$

$\quad F^a_{bias,i+1} = P^A_{i+1} F^A_{i+1} + H^A_{i+1} S_{i+1} D^{-1}_{i+1} \tau_{i+1} + H^a_{i+1} \ddot{X}_{bias,i+1};$

$\quad F^A_{bias,i} = F^A_{bias,i} + \sum {}^iX^*_{i+1} F^a_{bias,i+1};$

$\quad A_i = {}^iX^T_{i+1} P^A_{i+1} A_{i+1};$

$\quad U_i =$
$\quad\quad U_{i+1} + A^T_{i+1}\left\{\ddot{X}_{bias,i+1} + S_i D^{-1}\left(\tau_{i+1} - S^T_i (F^A_{bias,i+1} + H^a_{i+1}\ddot{X}_{bias,i+1})\right)\right\};$

$\quad \mathcal{L}_i = \mathcal{L}_{i+1} - A^T_{i+1} S_{i+1} D^{-1}_{i+1} S^T_{i+1} A_{i+1}$

end

/* *Linear constraint force magnitudes* */

$\nu = \mathcal{L}_0^{-1}(b_N - A^T_0 \ddot{X}_0 - U_0);$

/* *Outward sweep of acceleration* */

for $i \leftarrow 0$ *to* $N-1$ **do**

$\quad \ddot{q}_{i+1} = D^{-1}_{i+1}\left\{\tau_{i+1} - S^T_{i+1}\left(F^A_{bias,i+1} + H^A_{i+1}\left({}^{i+1}X_i \ddot{X}_i + \ddot{X}_{bias,i+1}\right) + A_{i+1}\nu\right)\right\};$

$\quad \ddot{X}_{i+1} = {}^{i+1}X_i \ddot{X}_i + \ddot{q}_{i+1} S_{i+1} + \ddot{X}_{bias,i+1};$

end

end

The required inputs to the algorithm (1) are listed below;

- *Robot model parameters* - A complete robot model defined by rigid body parameters such as mass, inertia, link lengths of individual segments.

- *Joint positions* defined at current time instance (q_i).
- *Joint velocities* ($\dot{q}_i$)
- *Feed-forward joint torques* (τ_i)
- *Cartesian acceleration* at current instance of time defined at the base ($\ddot{X}_0$).
- *External forces* (F_i^{ext}).
- *Unit constrained forces* applied at the end-effector defined as a matrix (A_N).
- *Acceleration energy set-point* defined at the end-effector (b_N).

In the following subsections, the associated equations are illustrated.

3.2.1 Outward sweep : position, velocity and acceleration recursions

The outward recursion solves the forward dynamics problem. In the algorithm, the first *for loop* iterates from the segment 0 (base) to segment $N-1$ (end-effector). During the recursion, it computes the pose, velocity and acceleration values. Furthermore, it calculates the bias forces and initializes rigid body inertia of the kinematic chain.

To compute the desired quantities two operations are used [8], one is the *change in reference point*, where the vector representation of the physical entities such as position, velocity and acceleration are changed from proximal joint pose frame $\{p_i\}$ to distal joint pose frame $\{d_i\}$, and other is incorporation of these entities with respect to joint $\{i+1\}$.

The pose from the segment i to $i+1$ is denoted as ${}^{i+1}_{i}X$. This is calculated by the combined transformation between proximal and distal segment frames attached to link (refer to figure 3.2.1). The two transformation matrices are,

- ${}^{d_i}_{i}X$ - pose from current segment i to distal pose frame d_i and;

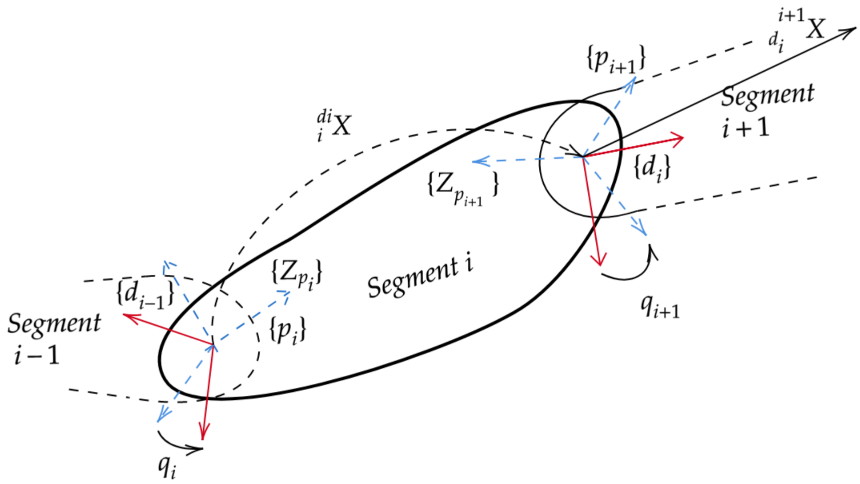


Figure 3.1: Proximal and distal segment frames attachment in a generic kinematic chain and transformation between them (${}^{d_i}_{i}X$) [8].

- ${}^{i+1}_{d_i}X$ - pose transformation of distal segment d_i to segment $i+1$.

In line 4, the *spatial velocity vector* at segment $i+1$ is calculated, which is represented by $\ddot{X}_{i+1}$. The expression is evaluated as the summation of ${}^{i+1}X_i\ddot{X}_i$ and $S_{i+1}\dot{q}_{i+1}$ recursively. Here the first term represents velocity of segment i expressed in the coordinates of segment $i+1$. The transformation from link i to $i+1$ is computed by matrix ${}^{i+1}X_i$. The second term refers to joint velocity contributions ($\dot{q}_{i+1}$) that is expressed using motion subspace matrix (S_{i+1}).

The next equation (line 5) denotes *bias acceleration* at segment $i+1$, noted as $\ddot{X}_{bias,i+1}$. Since the joint acceleration components are unknown at this stage, only the bias acceleration is computed, provided the cartesian and joint space acceleration of previous link. Here, $\dot{X} \times S$ signifies time derivative of S.

Furthermore, *bias forces* are determined by the expression in line 6, given the cartesian velocity vector, $\dot{X}_{i+1}$ and inertia matrix, H_{i+1}. The term $\times^*$ is the cross product operator expressed in Plücker coordinates (refer to appendix section C for explanation on spatial cross products). The bias forces is influenced by the *external forces* as well, given by $F^{ext}_{0,i+1}$ and is transformed from base to end-effector coordinates, expressed by transformation matrix for force vectors, ${}^{i+1}X^*_0$. See the appendix section D for coordinate transformation on force and motion vectors.

In the line 7 and 8, *articulated body inertia* and *articulated bias forces* respectively are initialized with *rigid body* quantities. These values are further used in inward sweep.

3.2.2 Inward sweep : force and inertia recursions

A set of recursive equations in inward sweep computes force and inertial parameters of every segment. The joint torques and external forces acting on the distal segments collectively generates *inertia-dependent acceleration* on the proximal segments [8].

In line 11, the combined inertias of segment $i+1$ and joint rotor inertia (d_{i+1}) is computed. Matrix P^A_{i+1} is a projection matrix, that projects *articulated body inertia* and *bias forces* to joint subspace [8] [11]. In further steps, the algorithm calculates *apparent inertia* (line 13) represented as H^a_{i+1}, which is the inertia contributions from the child segments. And *articulated body inertia* (line 14) denoted as H^A_i is calculated by adding all the apparent inertias. Similarly, apparent ($F^a_{bias,i+1}$) and articulated bias forces ($F^A_{bias,i}$) are computed by expression in line 15 and 16 respectively.

In expression 17, *constraint force matrix* is computed (A_i), in which the term $P^A_{i+1}A_{i+1}$, represents apparent unit constraint forces. Consequently, these constraint forces, external forces and joint torques inclusively generates acceleration energy [8], which is recursively accumulated in matrix U_i (line 18). Here, U_i is *acceleration energy* matrix expressed in Cartesian space. The expression within curly braces denotes acceleration originated from joint torques, and inertial forces applied at distal joints [8].

The inward recursion also deals with "*how much of the constraint acceleration energy b_N is already generated by each of the virtual "unit" constraint forces A_N* [8]". This is represented by $\mathcal{L}_i$ (line 19) and is called *constraint coupling matrix*, which is of order $m \times m$ (m is number of constraints). More clearly, each rows in $\mathcal{L}_i$ corresponds to acceleration energy generated by the all the constraint forces up until that instance of recursion.

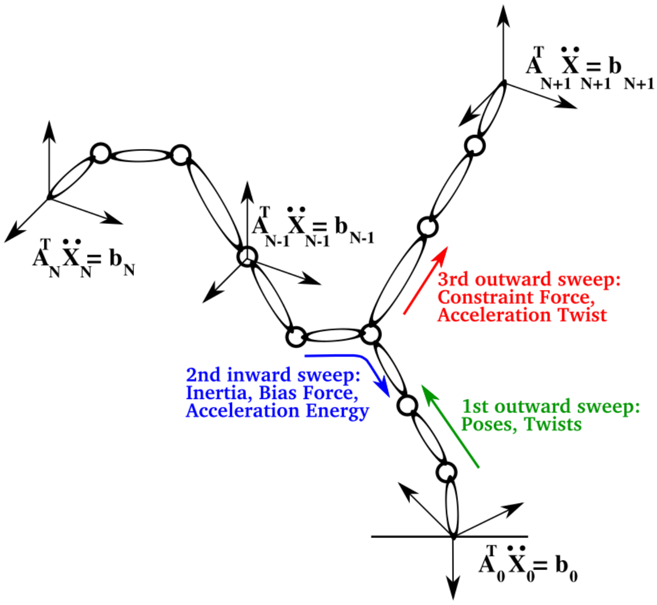


Figure 3.2: An abstract representation of computational sweeps in a kinematic chain, along with computed physical entities and constraints (source: [8])

.

3.2.3 Computing constraint force magnitudes, ν

After reaching the base ($i = 0$), the *constraint force magnitudes* is calculated (line 21). This expression is obtained after minimizing the Equation 3.5 with respect to ν [8]. The term $\ddot{X}_0$ denotes the cartesian acceleration at the base. Since the base is rigidly fixed in a kinematic chain, $\ddot{X}_0$ is equal to acceleration due to gravity.

It is however important to ensure that the matrix $\mathcal{L}_i$ is of full rank. But this case fails during the redundancy problem. To overcome this situation, ($\mathcal{L}^{-1}$) can be computed using the *damped least squares* method, as mentioned in [8].

3.2.4 Outward sweep : Control torques and link accelerations

In the outward sweep, the control torques and joint accelerations to the constrained motion are computed (line 23 and 24) [8]. After minimizing the equation

3.5 with respect to ν in previous step, the *constraint force magnitudes* is substituted and solved for joint acceleration $\ddot{q}_i$ in the final outward sweep. As mentioned before, the joint $i+1$ experiences external and Coriolis forces ($F^A_{bias,i+1}$), inertial forces ($H^A_{i+1}\,{}^{i+1}X_i\ddot{X}_i$) and feed-forward torques (τ_{i+1}) from the connected child segments. Corresponding to these quantities, the right hand side equation (line 23) represents the overall control torque that is required to drive the constrained system [11].

Reformulating the expression in line 23 and representing the torque components (see equation 3.7) [11],

$$\ddot{q}_{i+1} = D^{-1}_{i+1}\{ \overbrace{\tau_{i+1}}^{\text{input torque}} - \underbrace{S^T_{i+1}\big(F^A_{bias,i+1} + H^A_{i+1}\big({}^{i+1}X_i\ddot{X}_i + \ddot{X}_{bias,i+1}\big)\big)}_{\text{bias torque}} - \overbrace{S^T_{i+1}A_{i+1}\nu}^{\text{constraint torque}} \} \tag{3.7}$$

In the final step of the algorithm,

3.3 Acceleration constrained dynamics algorithm

3.4 Conclusion

4 Extending the Vereshchagin hybrid dynamic solver to mobile robots

Your main contributions go here

4.1 Extended algorithm

....

4.2 Implementation details

....

5

Evaluation

Implementation and measurements.

6

Results

6.1 Use case 1

Describe results and analyse them

6.2 Use case 2

6.3 Use case 3

7

Conclusions

7.1 Contributions

7.2 Lessons learned

7.3 Future work

A

Dynamic equation of motion

The general dynamic equation of motion of a rigid body is expressed as [5] [8],

$$M(q)\ddot{q} + C(q, \dot{q}) = \tau \tag{A.1}$$

where, $M(q)$ represents mapping from motion domain(M^n) to force domain(F^n). $C(q, \dot{q})$ is the Coriolis and Centrifugal forces acting on the rigid body. Both these quantities are dependent on q, $\dot{q}$, $\ddot{q}$ and the physical model of rigid body [5]

The dynamics problem is divided into forward and inverse dynamics. Computing the acceleration($\ddot{q}$), given the input forces(τ) is termed as *forward dynamics* problem. Conversely, *Inverse dynamics* problem calculates the forces, τ given accelerations $\ddot{q}$.

The rigid body is generally subjected to various motion constraints that changes the form of the dynamics equation. The extended equation is given by [8],

$$M(q)\ddot{q} = \tau_a(q) - \tau_c(q) - C(q, \dot{q}) \tag{A.2}$$

In the above equation, τ_a represents input forces and τ_c is the constraint forces from the task specification.

B

Plücker Notations for Spatial vectors

Your second chapter appendix

C

Plücker Notations for Spatial cross products

There are two spatial cross product operators expressed using Plücker notations are, $\times$ and $\times^*$ [5]. The operators can be regarded as dual to each other. The matrix representation of these operators are deduced as [5],

$$\hat{v}_O \times = \begin{bmatrix} \omega \\ v_O \end{bmatrix} \times = \begin{bmatrix} \omega\times & 0 \\ v_O\times & \omega\times \end{bmatrix} \tag{C.1}$$

and,

$$\hat{v}_O \times^* = \begin{bmatrix} \omega \\ v_O \end{bmatrix} \times^* = \begin{bmatrix} \omega\times & v_O\times \\ 0 & \omega\times \end{bmatrix} \tag{C.2}$$

D

Representation of coordinate transforms

In this section, the notations used to represent coordinate transformation matrices on motion and force vectors is presented. This follows the convention given in [5].

- ${}^{i+1}X_i$ - denotes coordinate transform from i to $i+1$ coordinates of a motion vector,
- ${}^{i+1}X_i^*$ - denotes coordinate transform from i to $i+1$ coordinates of a force vector.

These transforms are related by the following equation [5],

$$ {}^{i+1}X_i^* = {}^{i+1}X_i^{-T} $$

References

[1] Richard Bellman. *Dynamic programming.* Courier Corporation, 2013.

[2] Dimitri P Bertsekas. *Constrained optimization and Lagrange multiplier methods.* Academic press, 2014.

[3] Dimitri P Bertsekas et al. *Dynamic programming and optimal control*, volume 1. Athena Scientific Belmont, Massachusetts, 1996.

[4] Herman Bruyninckx and Oussama Khatib. Gauss' principle and the dynamics of redundant and constrained manipulators. In *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, volume 3, pages 2563–2568. IEEE, 2000.

[5] Roy Featherstone. *Rigid body dynamics algorithms.* Springer, 2014.

[6] Dieter Fox, Wolfram Burgard, and Sebastian Thrun. The dynamic window approach to collision avoidance. *IEEE Robotics & Automation Magazine*, 4(1): 23–33, 1997.

[7] Ekkehard Ramm. *Principles of Least Action and of Least Constraint*, pages 23–43. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014. ISBN 978-3-642-39905-3. doi: 10.1007/978-3-642-39905-3_2. URL https://doi.org/10.1007/978-3-642-39905-3_2.

[8] Azamat Shakhimardanov. Composable robot motion stack: Implementing constrained hybrid dynamics using semantic models of kinematic chains. 2015.

[9] Dongjun Shin, Irene Sardellitti, and Oussama Khatib. A hybrid actuation approach for human-friendly robot design. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 1747–1752. IEEE, 2008.

[10] AF Vereshchagin. Modeling and control of motion of manipulational robots. *SOVIET JOURNAL OF COMPUTER AND SYSTEMS SCIENCES*, 27(5): 29–38, 1989.

[11] Djordje Vukcevic. Extending a constrained hybrid dynamics solver for energy-optimal robot motions in the presence of static friction. *Technical Report/Hochschule Bonn-Rhein-Sieg-University of Applied Sciences, Department of Computer Science*, 2018.