Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

b-it Bonn-Aachen International Center for Information Technology

R&D Project

Exploiting contact constraints in robotic manipulation tasks

Wing Ki Lau

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. Nico Hochgeschwender
Sven Schneider, M.Sc.

September 2023

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.

12.09.2023

Date

Wing Ki Lau

Abstract

During the process of motion planning, robot manipulators often view contact surfaces as obstacles rather than opportunities. This approach stands in contrast to human behavior, where people frequently rest their wrists on a table while writing on paper. To enable robots to emulate this behavior, there exists a dynamic solver capable of addressing such requirements. The Popov-Vereshchagin hybrid solver allows for partial motion specifications in certain directions, allowing them to be influenced by natural forces. However, this solver remains relatively obscure within both academic and industrial circles, and there is limited research and implementation related to this topic.

The objective of the research and development project is to investigate the impact of robots leveraging environmental constraints. Three experiments were conducted to assess the energy efficiency and motion accuracy of robot manipulation when assigned to everyday tasks such as writing.

In the evaluation, it was observed that the average energy consumption and joint torque of most of the robot's joints decreased when performing a grasping task. Interestingly, the accuracy of motion in contact scenarios was found to be lower than in contactless situations. This discrepancy may be attributed to the need to overcome friction with the contact surface, thereby opening up opportunities for further research into the exploitation of contact constraints across different textures and shapes of contact surfaces. Additionally, it explores how different tasks affect the energy efficiency of leveraging contact surfaces during robot manipulation tasks.

Contents

List of Figures

List of Tables

1

Introduction

In the field of industrial robotics, achieving high levels of accuracy, precision, and repeatability in free-space motion is a critical requirement. To meet this requirement, robots must be designed to be stiff and avoid contact along their trajectories. One approach to achieving a high degree of stiffness is to use lightweight materials such as carbon fibre. However, the production of carbon fiber is expensive, leading some companies to opt for cheaper and heavier robot manipulators to reduce manufacturing costs. While this approach may reduce costs, it also results in increased energy consumption as the robot must expend more energy to maintain its joints and links in mid-air.

During motion planning, robot manipulators commonly treat contact surfaces as obstacles rather than opportunities. This approach contrasts with the behaviour of humans. Take a writing task as an example, as human we often rest our wrists on a table while writing on paper as figure 1.1b shows, however, robot tends to maintain their end effector in midair as figure 1.1a shows. By exploiting contact surfaces, robots can achieve selective improvements in accuracy and precision compared to writing on paper with their wrists in mid-air. Furthermore, utilising contact surfaces can help reduce energy consumption.

(a) The example of robot performing writing [1]

(b) Picture of human writing [2]

Figure 1.1: The comparison of robot and human performing writing task

To enable robots to imitate this behaviour, two adaptations of control software are required: Dynamics and partial constraint. Contact handling requires not only the kinematics of the robot but also its dynamics. There are existing dynamic solvers that can be used to solve the robot's equations of motion.

To fully exploit the contact surface, the specification of partial constraints must be introduced. The acceleration-constrained hybrid dynamics algorithm (ACHDA) limits the ability to handle the partial constraint of an end-effector. ACHDA can handle partial motion specifications in some direction and leave them determined by nature. Since nature determines part of the motion, the solver provides higher energy efficiency, passive adaptation, and desire constraints that do not require explicit control. Shakhimardanov has demonstrated the potential for extending constraints to arbitrary links [3]. However, this extension has not yet been incorporated into any software library. This means that while the theoretical framework for extending constraints to arbitrary links has been established, it has not yet been implemented in practice

In this research and development project, we will address three case studies to demonstrate a robot that exploits environmental constraints: Grasping an object by sliding along its surface, performing a writing task, and resting robot manipulation. The first case study aims to establish the required infrastructure for task execution through a proof-of-concept task. The second case study aims to implement the extension of the Vereshchagin solver in daily tasks to investigate whether exploiting contact surfaces can improve the accuracy and precision of robot motion. The third case study aims to evaluate whether resting some of the joints on a supporting surface can lead to higher energy efficiency.

The content of this research and development report is presented as follows. First, the State of the art will be introduced in chapter 3 to deliver a brief idea on the related work in dynamics, controllers, a stack of tasks, and some related work. From the deficits of the existing state of the art, chapter 4 will describe the core of this research and development project, Popov-Vereshchagin (PV) solver, especially the task interface of the PV solver. In the software development side, KDL library provides a function to aid the computation of differences between a variety kinds of data structures by using function overloading. After giving a concrete idea of the background of the PV solver and the robot interface, in chapter 5, robot architecture is proposed. It consists of two major parts: a cascaded controller to control the β value mentioned in 4 and the PV solver. Then the extension of Robif2b, a robot control interface, is introduced as it shows the flexibility such that we can selectively choose what port we can communicate with the robot. Then the experiment design, setup, and evaluation will be delivered in chapter 6 on a use case basis where three use cases: (1) Grasp object by sliding motion along surface (2) Perform writing task and (3) Resting elbow manipulation will be investigated. In the end, a conclusion will be given, to sum up the project and proposed possible future work.

2

Problem Statement

Many of today's robots are characterized by their substantial weight, high energy consumption, and expensive manufacturing costs. In the context of motion planning, contact surfaces are frequently regarded as hindrances, resulting in the robot's joints being suspended in mid-air. This approach can lead to wasteful energy usage as the robot expends energy to maintain these joints and links in such a suspended state. Furthermore, current dynamic solvers like the Articulated Body Algorithm (ABA) and the Recursive Newton-Euler Algorithm (RNEA) are ill-suited for handling partial constraint conditions. The Popov-Vereshchagin hybrid solver, although capable of accommodating such task specifications, remains relatively unknown within the robotics community, with a lack of comprehensive studies and implementations.

As humans, we go about our daily tasks by resting our arms on surfaces. Taking inspiration from this common human behaviour and the unique characteristics of the Popov-Vereshchagin hybrid solver, it becomes conceivable that robots could replicate such motions.

In the scope of our research and development project, we intend to explore an innovative approach that addresses the dynamics of this scenario. By re-framing contact surfaces as opportunities rather than obstacles, our objective is to cultivate a robotic motion planning solution that is not only more energy-efficient but also more cost-effective.

3

State of the Art

3.1 Robot dynamic

A robot manipulator, refers to a mechanical system consisting of a set of rigid bodies call *links* which interconnected by *joints*. These joints facilitate relative motion between neighbouring links, granting the manipulator the capability to maneuver in a variety of ways within its workspace. The spatial arrangement of the manipulator's joints and links lead to physical constraint which limits the direction of the motion that is being executed.
The motion of a rigid body, to be precise, the forces and accelerations of a rigid body, are being studied in the field of robot dynamics which encompasses both *forward* and *inverse* dynamics. The motions are described by the dynamic equation which being evaluated by dynamic algorithm which solve the numerical value that related to the dynamics [4]. In *forward Dynamic*(FD), an applied force to a rigid body is given to calculate the acceleration respond of such input [4]. The equation of motion of FD is expressed in the following form:

$$\ddot{q} = H(q)^{-1}(\tau - C(q, \dot{q})) \tag{3.1}$$

In equation (3.1), τ is a vector of generalised force in joint space. q , $\dot{q}$ and $\ddot{q}$ are the vector of position, velocity and acceleration in joint space respectively. $H(q)$ is the interia matrix which is a function of q. At last, $C(q, \dot{q})$ represent the generalised bias force which comprises Coriolis, centrifugal and gravity forces in joint space with other external force that possibly act on the system [4].
Conversely, Inverse dynamics (ID) computes the necessary force required to generate a given or desired acceleration within a system of rigid bodies. The equation of motion of ID is expressed in the following form [4]:

$$\tau = H(q)\ddot{q} + C(q, \dot{q}) \tag{3.2}$$

Besides forward dynamic and inverse dynamic, Hybrid Dynamic (HD) contains both dynamic problems by given/known τ or $\ddot{q}$ of a specific joint to compute the unknown forces and accelerations.
Practically, as a safer method to visualise the effect of validating the solution of control signal and power consumption,in research and development,researchers and roboticitsts frequently use stimulation to

perform experiments and developments before deploying on a physical robot. There are many existing software solutions and libraries provide physics engine solutions for simulating physical interactions and dynamics in robotics. Dynamic Animation and Robotics Toolkit (DART) [5], Open Dynamics Engine (ODE) [6], Bullet Real-Time Physics engine [7] and MuJoCo (Multi-Joint dynamics with Contact) physics engine [8]. The physics engines mentioned above uses numbers of aforementioned dynamic solvers such as The Recursive Newton-Euler Algorithm (RNEA) and Articulated Body Algorithm (ABA).

3.2 Dynamic solver

A dynamic solver refers to a software or algorithmic element tasked with resolving dynamic equations of motion. These equations elucidate how objects, whether they be robot manipulators or simulated physical entities, undergo motion and engage with the forces and torques exerted upon them. This session introduces some common dynamic algorithms that are being used to solve the dynamic problem mentioned aforenamedly.

The Articulated Body Algorithm (ABA)

An articulated rigid body system consists of interconnected rigid bodies (links) connected by joints. Each rigid body has mass, inertia, and geometry, and each joint provides relative motion between adjacent bodies. The Articulated Body Algorithm (ABA) computes the dynamics and motion of articulated bodies. This algorithm models the dynamics of each body in the system using a hierarchical approach, taking into account the interactions between the bodies and the forces acting upon them [9], [10]. The ABA solver is used to solve a forward dynamics problem, where accelerations are calculated from input torques [10].

Algorithm 1: The Articulated Body Algorithm (ABA) [4]

$\boldsymbol{v_0} = \mathbf{0}$
begin
  for *i = 1 to* N_B **do**
    $[\mathbf{X}_j, \mathbf{S}_i, \boldsymbol{v}_J, \boldsymbol{c}_J]$ =jcalc(jtype(i),$\boldsymbol{q_i}$,$\dot{\boldsymbol{q}}_{\boldsymbol{i}}$)
    ${}^{i}\mathbf{X}_{\lambda(i)} = \mathbf{X}_J \mathbf{X}_T(i)$
    if $\lambda(i) \neq 0$ **then**
      ${}^{i}\mathbf{X}_0 = {}^{i}\mathbf{X}_{\lambda(i)}{}^{\lambda(i)}\mathbf{X}_0$
    end
    $\boldsymbol{v}_i = {}^{i}\mathbf{X}_{\lambda(i)} \boldsymbol{v}_{\lambda(i)} + \boldsymbol{v}_j$
    $\boldsymbol{c}_i = \boldsymbol{c}_J + \boldsymbol{v}_i \times \boldsymbol{v}_j$
    $\boldsymbol{I}_i^A = \boldsymbol{I}_i$
    $\boldsymbol{p}_i^A = \boldsymbol{v}_i \times^* \boldsymbol{I}_i \boldsymbol{v}_i - {}^{i}X_0^* f_i^x$
  end
  for *i =* N_b *to 1* **do**
    $\boldsymbol{U}_i = I_i^A S_i$
    $\boldsymbol{D}_i = S_i^T U_i$
    $\boldsymbol{u}_i = \tau_i - \boldsymbol{S}_i^T \boldsymbol{p}_i^A$
    if $\lambda(i) \neq 0$ **then**
      $\boldsymbol{I}^a = \boldsymbol{I}_i^A - \boldsymbol{U}_i \boldsymbol{D}_i^{-1} \boldsymbol{U}_i^T$
      $\boldsymbol{p}^a = \boldsymbol{p}_i^A + \boldsymbol{I}^a \boldsymbol{c}_i + \boldsymbol{U}_i \boldsymbol{D}_i^{-1} \boldsymbol{u}_i$
      $\boldsymbol{I}_{\lambda(i)}^A = \boldsymbol{I}_{\lambda(i)}^A + {}^{\lambda(i)}\boldsymbol{X}_i^* \boldsymbol{I}^{a\,i} \boldsymbol{X}_{\lambda(i)}$
      $\boldsymbol{p}_{\lambda(i)}^A = \boldsymbol{p}_{\lambda(i)}^A + {}^{\lambda(i)}\boldsymbol{X}_i^* \boldsymbol{p}^a$
    end
  end
  $\boldsymbol{a}_0 = -\boldsymbol{a}_g$
  for *i = 1 to* N_B **do**
    $\boldsymbol{a}' = {}^{i}\boldsymbol{X}_{\lambda(i)} a_{\lambda(i)} + \boldsymbol{c}_i$
    $\ddot{\boldsymbol{q}}_{\boldsymbol{i}} = \boldsymbol{D}_i^{-1}(\boldsymbol{u}_i - \boldsymbol{U}^T \boldsymbol{a}')$
    $\boldsymbol{a}_i = \boldsymbol{a}' + \boldsymbol{S}_i \ddot{\boldsymbol{q}}_i$
  end
end

In the context of the Articulated Body Algorithm (ABA) and a robot with 6 degrees of freedom (DOF), ABA does not support the definition of partial task specifications for bias forces that are not represented as a 6x1 vector. As an illustration, consider the scenario where we aim to define a bias force along the linear z-direction, which naturally arises from forces like gravity. Achieving this particular specification can prove challenging within the Articulated Body Algorithm (ABA) because, in a 6x1 vector, we can only assign a zero value to such a direction. However, merely setting a value to zero does not faithfully replicate

the motion dictated by nature. In contrast, the Popov-Vereshchagin hybrid solver offers a solution. As outlined in Chapter 4.2, it allows for the assignment of a 5x1 vector, enabling partial specification and providing the flexibility to accurately capture the desired behaviour.

The Recursive Newton-Euler Algorithm (RNEA)

The Recursive Newton-Euler Algorithm (RNEA) is a recursive approach to solving the inverse dynamics problem, which involves calculating joint-space torques from joint-space accelerations for each body in a system. This algorithm takes into account the relationships between the bodies and the forces acting upon them. It is aimed to streamline and improve the computation of dynamics in complex multi-segmented robot manipulator. The recursive approach minimises repetitive calculations and boosts computational effectiveness.Featherstone's book illustrate the algorithm as follow: [4].

Algorithm 2: The Recursive Newton-Euler Algorithm (RNEA)

1 $\boldsymbol{v_0} = \boldsymbol{0}$
2 $\boldsymbol{a_0} = \boldsymbol{0}$
3 **begin**
4 **for** $i = 1$ *to* N_B **do**
5 $[\mathbf{X}_J, \mathbf{S}_i, \boldsymbol{v}_J, \boldsymbol{c}_J]$ =jcalc(jtype(i),$\boldsymbol{q_i}$,$\dot{\boldsymbol{q}}_{\boldsymbol{i}}$)
6 ${}^i\mathbf{X}_{\lambda(i)} = \mathbf{X}_J\mathbf{X}_T(i)$
7 **if** $\lambda(i) \neq 0$ **then**
8 ${}^i\mathbf{X}_0 = {}^i\mathbf{X}_{\lambda(i)}{}^{\lambda(i)}\mathbf{X}_0$
9 **end**
10 $\boldsymbol{v}_i = {}^i\mathbf{X}_{\lambda(i)}\boldsymbol{v}_{\lambda(i)} + \boldsymbol{v}_j$
11 $\boldsymbol{a}_i = {}^i\mathbf{X}_{\lambda(i)}\boldsymbol{a}_{\lambda(i)} + \mathbf{S}_i\ddot{q}_i + \boldsymbol{c}_J + \boldsymbol{v}_i \times \boldsymbol{v}_J$
12 $\boldsymbol{f}_i = \boldsymbol{I}_i\boldsymbol{a}_i + \boldsymbol{v}_i \times^* \boldsymbol{I}_i\boldsymbol{v}_i - {}^i\mathbf{X}_0^*\boldsymbol{f}_i^x$
13 **end**
14 **for** $i = N_b$ *to* 1 **do**
15 $\tau_i = \boldsymbol{S}_i^T\boldsymbol{f}_i$
16 **if** $\lambda(i) \neq 0$ **then**
17 $\boldsymbol{f}_{\lambda(i)} = \boldsymbol{f}_{\lambda(i)} + {}^{\lambda(i)}\mathbf{X}_i^*\boldsymbol{f}_i^x$
18 **end**
19 **end**
20 **end**

Notice on line 11, the algorithm takes $\ddot{q}_i$ as input to compute a_i. However when we input a task specification or constraint $\ddot{\boldsymbol{X}}_N$, we need to map the task specification in Cartesian space $\ddot{\boldsymbol{X}}_N$ to joint space $\ddot{q}_i$ with:

$$\ddot{\boldsymbol{X}}_N = \boldsymbol{J}_N\ddot{q} \tag{3.3}$$

$$\ddot{q} = \boldsymbol{J}_N^{-1}\ddot{\boldsymbol{X}}_N \tag{3.4}$$

From 3.4 , $\ddot{\boldsymbol{X}}_N$ requires task specification in all 6 directions which lead to lacking flexibility of giving particle task specification. However, Popov-Vereshchagin hybrid solver resolves this issues by allowing particle task specification. A details explanation of different task specification will be discussed in chapter 4 section 4.2.

The Articulated-Body Hybrid Dynamics Algorithm (ACHDA)

The Articulated-Body Hybrid Dynamics Algorithm (ACHDA) is capable of computing three types of dynamics: inverse dynamics, forward dynamics, and hybrid dynamics. In inverse dynamics, the solver calculates joint torques from Cartesian acceleration constraints. In forward dynamics, the solver takes feed-forward joint torques as input and computes Cartesian accelerations. In hybrid dynamics, the solver combines both of the above methods. Additionally, as a byproduct, it can also take external forces as input and compute joint accelerations. [4].

3.3 Controller

Controller guides systems toward desired set-points or references value with feedback loop. The followings are some common controllers that are widely used in robotics.

Proportional-integral-derivative (PID) controllers

Proportional-Integral-Derivative (PID) controllers uphold a specific set-point. In this R&D project, PID controller regulates position and velocity by using the equation below:

$$u(t) = K_p e(t) + K_i \sum_{t=0}^{t} e(t)\Delta t + K_d \frac{e(t) - e(t-1)}{\Delta t} \tag{3.5}$$

Where $u(t)$ is the control signal at time t. $e(t)$ is the error at time t as the difference between the set-point and the current input of the controller. Δt is the time step of each calculation in the control loop. K_p, K_i, K_d are the coefficients of proportional, integral and derivative gain respectively [11].

Fuzzy logic algorithm

Fuzzy logic algorithms employed as controllers adeptly navigate scenarios marked by uncertainty, adeptly utilising linguistic terms and membership functions to accurately represent the gamut of inputs and outputs. This approach hinges upon rule-based decision-making, whereby the system gauges the extent of each rule's applicability contingent upon the values presented in the input domain [12].

Impedance controllers

Impedance controller emulate the the mechanical properties of stiffness and damping to regulate the forces and motions relationship when the robot interacts with objects. Extend from equation (3.2) with two main components: Stiffness and Damping Component. A basic Impedance controller is formed.

$$\tau = K(q_d - q) + D(\dot{q_d} - \dot{q}) + H(q)\ddot{q} + C(q, \dot{q}) \tag{3.6}$$

Where K and D are stiffness-like and damping-like matrix, respectively [13].

3.4 Task based control

Task-based control in robotics refers to a control approach that focuses on achieving specific tasks or objectives rather than directly controlling individual joints or degrees of freedom. Instead of commanding the robot's joints to move in a prescribed manner, task-based control involves specifying the desired behaviours or goals the robot should accomplish.

iTaSC

iTaSC is a control framework that was introduced by [14]. It is the fusion of instantaneous task specification and estimation of geometric uncertainty that focuses on achieving tasks while respecting various constraints. These constraints could include safety considerations, physical limitations, interaction forces, and more. The framework provides a way to dynamically adjust a robot's behaviour in real-time, allowing it to respond to changing conditions and maintain safe and effective interactions with the environment. Later [15] extended the idea of iTaSC.

In a multi-sensor robot system, a task is described as the desired behaviours or objectives that the robot needs to achieve. These tasks are formulated in the task space, which represents the robot's end-effector or operational space. The tasks and constraints had to be formulated at the beginning. Task formulation encompasses goal position, following a particular trajectory, or maintaining a certain force. constraint formulation contains obeying joint limits, velocity limits, and avoiding collisions. etc. Based on their priorities, tasks, and constraints are organized in a hierarchical manner. Two sets of coordinates were introduced to represent the tasks and constraints, feature, and uncertainty coordinates. Feature coordinate expresses the relative motions between features on specific objects. uncertainty coordinate represents geometric uncertainties.

The authors proposed a general control and estimation scheme: Where the robot system and the environment are denoted as *plant* P which observes the sensor measurements z. $u$is the control input (joint positions, velocities, and torques) that is computed from controller C. Controller takes the estimates X_u and $\widehat{y}$ as input where both of them are generated from *Model Update and Estimation block* M+E estimate $\widehat{y}$ which takes the controlled input signal u and measurements z as input to estimate y as the output of the system.

Later, [16] proposed using iTaSC on force-senseless robot force control. Figure 3.2 shows the one degree-of-freedom (DOF) robot force control scheme that can extend to multi-degree-of-freedom scenario.

In figure 3.2, the robot system model and the underlying joint velocity controller are depicted in red. The stiffness model pertaining to environmental contact is showcased in green, while the controller section open to design is highlighted in blue. The gravity term is omitted because the robot was being used in the experiment was gravity-compensated mechanically. This is a first order system with a desired input τ_d and output $\dot{q}$ in non-contact situation. In contrast,the green part is included. The model characterises a second-order system where τ_d serves as the input, and the resulting torque τ_w exerted on the environment

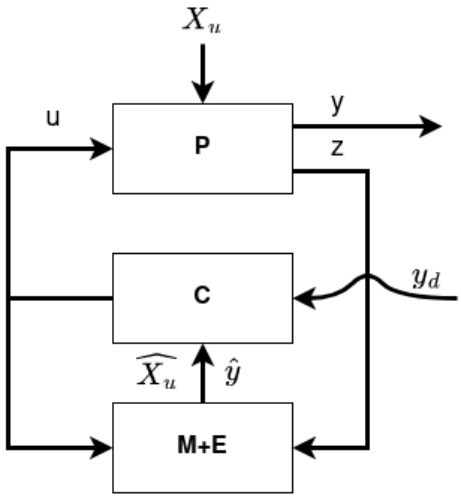


Figure 3.1: A general control and estimation scheme.
[14]

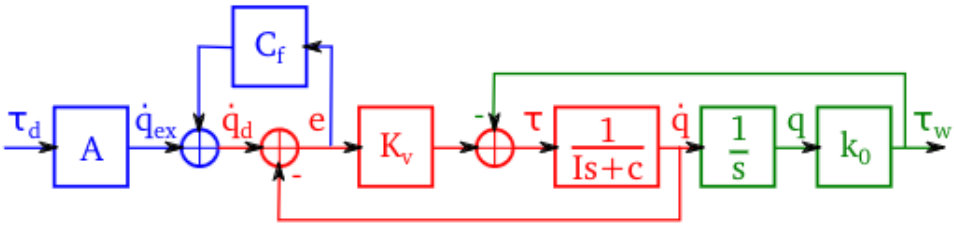


Figure 3.2: One degree-of-freedom robot force control scheme.
[16]

acts as the output. The joint velocity error feedback constant, C_f, results in the modification of the velocity loop feedback constant K_v, leading to an equated velocity loop feedback constant $\frac{K_v}{1-C_f}$. In multi degree-of-freedom scenario, all the previously mentioned variables becomes vector and the constants becomes matrices [16].

Stack of tasks

The paper from [17] and [18] introduced Stack of tasks (SoT) which provides a way to manage multiple objectives and constraints in a coherent and organised manner. It ensures that the robot's control algorithm focuses on achieving the most critical tasks first while considering the constraints and interactions among tasks and generalised the concept of hierarchy-based control strategies to include unilateral constraints at various levels of priority.

3.5 Existing implementation of exploiting surface in robot manipulation

There were some existing implementation or discussion about exploiting surface in robot manipulation. In [19] ,authors presented a mathematical framework that outlines the interaction between compliant hands and environmental constraints during the execution of a grasping task. [20] discussed planning grasp strategies that exploit environmental constraints. In this paper, each environmental constraint exploitation is considered as one controller with a desired spatial and contact condition with a termination predicate which is also a switching condition if the constraint is in between the motion in global point of view. The final motion planing consists of a series of environmental exploits that lead to a grasp. This paper [21] creates practical explanations of how forces and movements are interconnected in the context of sliding manipulation. This will be achieved by examining the concept of the limit surface, which originates from the mechanics of sliding bodies. Through kinematic analysis, we aim to determine the necessary force and torque to generate specific sliding motions.

3.6 Limitations of previous work

In today's world, there's an ongoing challenge in deciding between investing in an expensive, lightweight robot arm or a more affordable but heavier one. This decision-making is influenced by limited financial resources for development. Opting for cheaper, heavier robots can lead to smoother but less precise robotic movements, affecting accuracy.

Interestingly, current robot motion planning still treats contact surfaces as obstacles to be avoided. The well-known dynamic solvers like ABA and RNEA don't handle situations where only some constraints are present very well. As of now, the existing researches rely on visual or other sensory input to detect contact surface. However,there's a lack of research on how to adapt the Vereshchagin solver to scenarios where contact is made by joints other than the end-effector, and how to practically implement this idea without including sensors on the robot. The related work does not directly describe the relationship between the sliding and frictional components in a smooth model of the resultant friction force on a plane contact area [22]. Also with ACHDA, [23] notice the lack of singularity prevention may lead to physical damage of the robot manipulator and the influence of external forces to the system is neglected [23].

4

Background

In section 3.6, the limitations of the current research has been raised. In order to control the robot to move along a contact surface without sensory input, there were researchers, Vereshchagin and Popov, proposed a hybrid dynamic algorithms and tried to tackle the problem of exploiting the defined constraints,robot model, joint angles, joint velocities, feed forward torque and external forces to compute the required joint torque and joint accelerations as command that being sent to the robots [24], [25]. By taking the concept of power is produced by force act on a certain velocity, acceleration energy is produced by acting force on a certain acceleration.

4.1 Popov-Vereshchagin hybrid solver description

The solver builds from *Gauss principle of least constraints* , the basic principle of mechanics with a low linear time complexity of O(n). Gauss' principle asserts that the genuine motion (acceleration) of a system or body is determined by minimising a quadratic function under the conditions of linear geometric motion constraints [26], [27]. The outcome of this Gauss function signifies the "acceleration energy" associated with a body. This energy is calculated as the product of the body's mass and the squared difference between its permissible (constrained) acceleration and its unrestricted (unconstrained) acceleration [28]. The detailed description can be found in [24], [29] and [30]. To describe Vereshchagin solver in a simpler manner, the solver takes the following parameters as input:

- Robot's model, which defined by: kinematic parameters of the chain, segments' mass and rigid-body inertia, and effective inertia of each joint rotor
- Root acceleration of the robot's base segment
- Joint angles at the current time frame
- Joint velocities at the current time frame
- Feedfoward joint torques
- External force applies on the robot
- Cartesian Acceleration Constraints

- Set-points which measured in unit of acceleration energy

With the two parameters as output:

- The resulting joint accelerations defined at end-effector frame
- The resulting joint constraint torques defined at end-effector frame

The algorithm computes the required robot motion at the instantaneous time step based on the aforementioned input parameters with three computation sweeps. The detail explanation can refer to [24] , [31] and [32].

4.2 Task interfaces

One crucial feature or Popov-Vereshchagin hybrid solver is it can directly use task definitions as input so calculate the desire robot motions. There are three task interfaces can be specified in the input:

1. The virtual and physical external force acting on each segments of the robot F_{ext}
2. The feed-forward force τ acts on each joint
3. The Cartesian Acceleration Constraints that impose on the end-effector segment according the equation $\alpha_N^T \ddot{X}_N = \beta_N$

Task interfaces: F_{ext}

The initial type of task definition is suitable for describing physical forces and torques applied to each of the robot's segments in Cartesian coordinates, as opposed to artificial ones. For example, human acts force on robot segments or the extra weight from gripper.

Task interfaces: τ

The second type of task definition can be used for specifying physical joint torques, for example,the static spring and/or damper-based torques in robot's joints.

Task interfaces: $\alpha_N^T \ddot{X}_N = \beta_N$

The last type of task definition manages physical interactions with the environment, such as contact, and also deals with artificial constraints set by the operational space task description for the end-effector segment. The α_N^T of the interface is the active constraint directions which is a $6xm$ matrix with spatial unit constraint forces and β_N is a $mx1$ vector,where m = the number of constraint. Assume the number of constraint is 1 such that $m = 1$ and user wants to constraint the motion along the linear x direction at

the end-effector segment:

$$\alpha_N = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \qquad \beta_n = \begin{bmatrix} 0 \end{bmatrix} \tag{4.1}$$

In α_N matrix, the first three rows represent the linear elements and the last three rows represent angular elements respectively. By giving zero value to acceleration to energy set-point,the user is defining that the end-effector is not allowed to have linear acceleration in x direction. To put this in other words, the user give a constraint of the motion on end effector of the robot along linear x direction. The other example is to given constraints specification in 5 DOFs. User can constraint the end effector to move freely only along z direction without any rotation motion:

$$\alpha_N = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \qquad \beta_n = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{4.2}$$

In the 5 DOFs example, the constrained joint torques will be computed by the Acceleration Constrained Hybrid Dynamics (ACHD) solver even the linear z direction motion is not being specified. Unconstrained motion specifications are naturally resolved using Gauss' principle of least constraint. This means that directions in which the robot is not constrained by the task definition, its motions will be controlled by the nature. In this example, the end effector will fall along linear z direction due to the weight of the link and gravity.

The last example demonstrate a full task specification of the desired end-effector motion in all 6 DOFs, means there are total 6 constraints and $m = 6$ such that:

$$\alpha_N = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \qquad \beta_N = \alpha_N^T \ddot{X}_N \tag{4.3}$$

where N is the index of robot's end effector. We can directly assign the magnitudes of the desired/task-defined spatial acceleration $\ddot{X}_N$ to the 6 x 1 vector of acceleration energy β_N such that $\beta_N = \ddot{X}_N$. While

the physical dimensions (units) of these two vectors are dissimilar, the presence of unit vectors in the α_N property enables us to allocate desired acceleration values to energy set-points for acceleration in their respective directions.Specifically, every column within the α_N holds a value of 1 in the corresponding direction where constraint force is applied. As a result, the acceleration energy set-point value aligns with the Cartesian acceleration value in the corresponding direction. The original Popov-Vereshchagin solver only account for the end effector of the robot and recently, [33] extended the algorithm to impose constraints to all robot segments.

4.3 Calculate differences with KDL library

We need a controller to assign β_N according to a set-point. In order to compute the control signal, we need to calculate the difference between measured value and set-point value. The direct way to calculate position difference is to calculate the difference between current frame and target frame. First, divide a frame into linear part and rotation part. We compute the position linear difference with:

$$p_{err} = p_{measure} - p_{setpoint} \tag{4.4}$$

In rotation part:

$$M_{err,angular} = M_{measure,angular} * M^{-1}_{setpoint,angular} \tag{4.5}$$

where p is a 3 x 1 vector that represents the x, y and z linear direction and M is a rotation matrix.
To calculate velocities error, it is a direct computation:

$$V_{err} = V_{measure} - V_{setpoint}, \qquad V = \begin{bmatrix} v_{lin,x} \\ v_{lin,y} \\ v_{lin,z} \\ v_{ang,x} \\ v_{ang,y} \\ v_{ang,z} \end{bmatrix} \tag{4.6}$$

where V is a 6 x 1 vector which composes both linear and angular velocities.
There is a difficulties to present the physical meaning since the rotation error and control signal are in matrix form . When we use a cascaded controller, the position control signal will be feed into the velocities controller as a set-point. However, according to 4.5 and 4.6 , mapping a rotation matrix to a velocity vector poses a challenge. To address these issues, the KDL library offers various methods for calculating frame differences. The *diff()* function is overloaded for all classes in *frames.hpp* and *framesvel.hpp* according to difference data structure of parameters as input:

- Frame
- Rotation

- Twist
- Wrench

In this research and development project, we employ *Frame* and *Twist* data structure as parameters. A Frame class in KDL library represents a frame transformation (rotation and translation) in 3D space. It composes with a 3 x 1 Vector (translation) data structure p and a 3 x 3 matrix M with the Rotation data structure:

$$p = \begin{bmatrix} x \\ y \\ zz \end{bmatrix} \qquad M = \begin{bmatrix} X_x, Y_x, Z_x \\ X_y, Y_y, Z_y \\ X_z, Y_z, Z_z \end{bmatrix} \tag{4.7}$$

diff(Frame b1,Frame b2)

The first function overload allows *diff()* determines the rotation axis necessary to rotate the frame b1 to the same orientation as frame b2 and the vector necessary to translate the origin of b1 to the origin of b2, and stores the result in a Twist data structure. However, it's important to note that this *Twist* type value does not possess the characteristic properties of a typical twist.The output *Twist* data structure is a mixture of axis-angle Representation and translation Vector. In this report, this *Twist* data structure will be called *Pose Vector*
In general, the rotation vector representation is a compact way to describe a three-dimensional rotation or orientation in space and present how an frame is rotated from its original orientation to a new orientation without the complexities of matrices or quaternions.

diff(Twist a,Twist b)

The second function overload determines the difference between the velocity vectors and rotational velocity vectors. In this context since the Twist input consist the current linear and angular velocity of the end effector and set-point in world frame. The output of *diff(Twist_a,Twist_b)* in this context will be the velocity error between set-point and measured value from the robot end effector w.r.t to world frame.

5

Solution

We discussed various kinds of controllers in chapter 3 session 3.3 and the background knowledge of Popov-Vereshchagin hybrid solver in chapter 4.In this chapter, The proposed robot architecture session will present how the aforementioned solver and controller are implemented on the robot. Session extension of Robif2b will discuss how to modify Robif2b such that the communication between the robot and the software interface can be extended.

5.1 Proposed robot architecture

Figure 5.1 shows the robot system architecture majorly consists of a cascaded controller, the Popov-Vereshchagin solver, and the robot. In 4 mentioned the inputs of the Popov-Vereshchagin solver. The cascaded controller is composed of a P-position controller and a P-velocity controller. The main program fetches the instantaneous joint angle q from the robot, then performs forward kinematic to compute the current Cartesian pose of the robot's end-effector X_{curr}. Then calculate the position error term X_{err} by calling *diff*(X_{goal},X_{curr}) mentioned in chapter 4.3 where X_{goal} is the desired pose of the robot's end-effector of the current task. The P-position controller computes the position control signal $ctrl\_sig_{pos}$ and feeds into the P-velocity controller with the current Cartesian of the end-effector $\dot{X}_{curr}$. Before the main loop starts, $\dot{X}_{curr}$ is assumed to be zero in all linear and angular directions. After the first iteration, by calling function $getLinkCartesianPose()$ from the vereshchagin solver, it will update the $\dot{X}_{curr}$. The error is the difference between position control signal $ctrl\_sig_{pos}$ and $\dot{X}_{curr}$. P-velocity controller computes the control signal, which directly being set as acceleration energy set-points β_N in the Popov-Vereshchagin solver. In the software, we have two sets of P-gain for each P-controller $k_{pos} = \{k_{linear}, k_{angular}\}$ and $k_{vel} = \{k_{linear}, k_{angular}\}$ represent the P gain in linear and angular direction separately on each controller. These sets of P-gain values will vary according to the use case. In use case 1 , $k_{pos} = 40, 80$ and $k_{vel} = \{30, 2\}$ before establishing contact. After the end effector has contact with the table, the P-gain set changes to $k_{pos} = \{10, 80\}$ and $k_{vel} = \{4, 40\}$. This shows that implementing a cascaded controller required a different set of P-gain in contact and contactless cases.

We mentioned the inputs of the Popov-Vereshchagin solver in 4 and from above, joint angles and velocities q and $\dot{(q)}$ of the end-effector at the current time frame can be obtained from the robot. Feedforward joint torques remain zero since it is not necessary to propose task specification on joint

torques such as spring and/or damper-based torques in robot's joints [34]. The external forces driver F_{ext} will be set to a certain value according to the task specification. In the experiment,F_{ext} is set to some value to counter friction during a contact situation. The details will be further explained in the next chapter. As we discussed in 4.2, the structure of α_N depends on the task specification. For example, when the robot arm is moving in mid-air, the direction of both linear and angular motion should be constrained and in a contact situation, the constraint of linear z direction should be disabled. At last, β_N is being controlled by the cascaded controller that is aforementioned.

The solver requires the robot chain during declaration. There are two ways to implement such a robot chain. the first one is to create a robot chain class and use typical KDL robot declaration: *chain.addSegment()* to create the chain by referring to the Kinova robot manual. However, KDL expresses the rigid-body inertia in the link's tip frame whereas the manual chooses the link's root frame. Hence, inverse transformation is needed. Unfortunately, the frame assignment for the DH parameters in the manual is also encountering the same issue, so additional transformations of the inertia would be required when referring to the DH parameters in the user manual. To avoid such a problem, we are using kdl_parser library [35], [36]. KDL utilizes a tree structure to model and store the kinematic and dynamic characteristics of a robot mechanism. The kdl_parser offers utilities to create a KDL tree based on an XML robot description in URDF format Originally, this was a ROS package. However, as our project is not developed within the ROS framework, we have removed the dependencies on ROS.

The Popov-Vereshchagin solver has two properties: It does not support fixed joint and tree structure. So in kdl_parser, we have to create the chain until the bracelet links instead of the end-effector frame since the gripper has a tree structure. In order to construct the correct robot chain that includes the gripper, we have to add the weight of the gripper in the bracelet link in the urdf file such that the weight of the gripper is taken into account when computing robot dynamics [37].

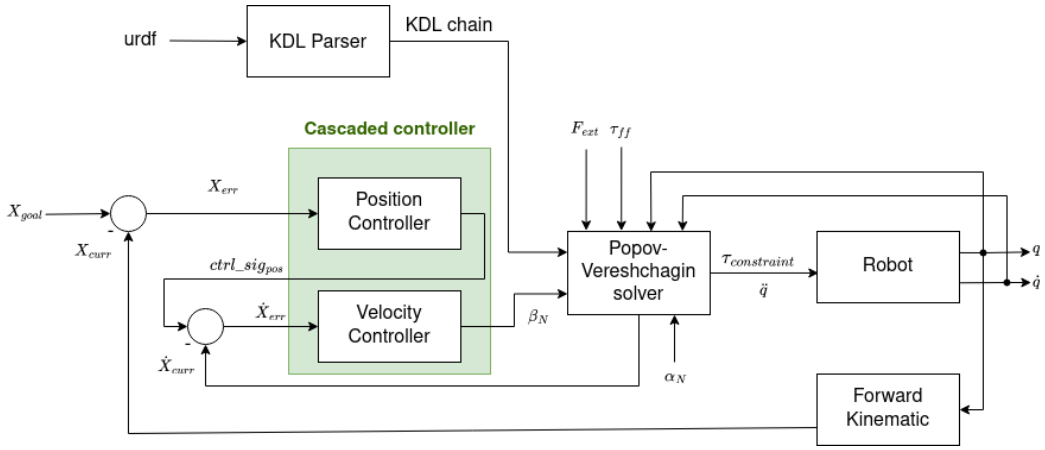


Figure 5.1: A generic robot system architecture

5.2 Extension of Robif2b

In this research and development project, the Kinova® Gen3 Ultra lightweight robot manipulator is employed, and a Kinova Kortex API client facilitates communication between the robot and the software [38], [39]. The script employs low-level servoing control as it circumvents the kinematic control library. This approach permits clients to independently control each actuator by transmitting command increments at a frequency of 1 kHz to increase the respond speed. Figure 5.2 show the connection between actuator, robot base and client in Low-level servoing mode. If we control robot with the Kortex api

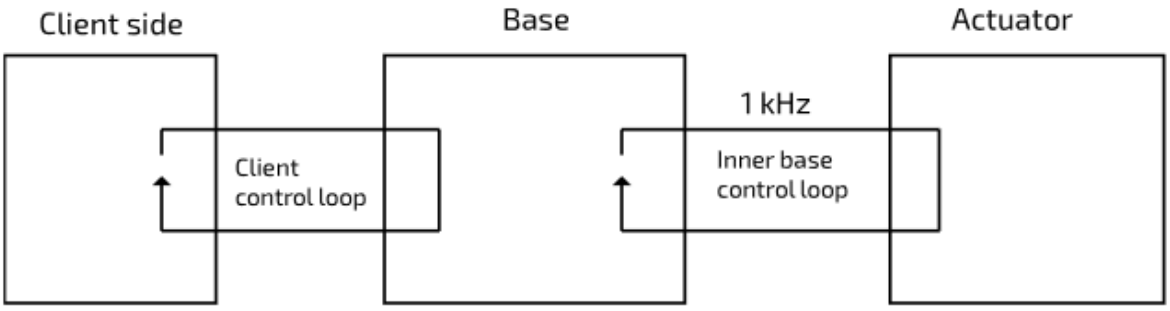


Figure 5.2: Connection between actuator,robot base and client in Low-level servoing

directly, we need to establish TCP/UDP connection and create session for data connection. Beside this complex implementation, there is another simple alternative. Robif2b is robot control interface wraps the session creation, connection establishment and communication between actuator, base and client side [40]. In the original Robif2b, the connection of actuator is already being establish to received the feedback from the base such that we can write a single line of command to receive the current status of base and actuators. Nonetheless, it didn't encompass the capability to retrieve the voltage and current status of the gripper actuator. The code snippet below demonstrates how the port is configured to receive current status updates for the base and actuators:

Listing 5.1: Struct of the enabled ports for base and actuator of Kinova kortex api in Robif2b

```
struct robif2b_kinova_gen3_nbx
{
    // Configuration
    struct robif2b_kinova_gen3_config conf;
    // Ports
    const double *cycle_time;                // [s]
    enum robif2b_ctrl_mode *ctrl_mode;
    double *jnt_pos_msr;                     // [rad]
    double *jnt_vel_msr;                     // [rad/s]
    double *jnt_trq_msr;                     // [Nm]
    double *act_cur_msr;                     // [A]
    double *act_vol_msr;                     // [V]
    double *gripper_pos_msr;
    const double *jnt_pos_cmd;               // [rad]
```

```
15          const double *jnt_vel_cmd;                  // [rad/s]
16          const double *jnt_trq_cmd;                  // [Nm]
17          const double *act_cur_cmd;                  // [A]
18
19          const double *gripper_pos_cmd;
20          double *imu_ang_vel_msr;                    // XYZ [rad/s]
21          double *imu_lin_acc_msr;                    // XYZ [m/s^2]
22
23          bool *success;
24          // Internal state
25          struct robif2b_kinova_gen3_comm *comm;
26          enum robif2b_ctrl_mode ctrl_mode_prev;
27      };
```

Line 12,13 and 19 are the extended port from original Robif2b such that we can retrieve the current voltage of each joint and the gripper position and we able to send the command to the gripper via sending the gripper position (0-100) with *gripper_pos_cmd*. We can extend this struct according to the kortex api manual [38] according to the need of evaluation. Before sending the command to gripper to receive its current state or control the gripper, we have to declare the connection to the base and gripper. The code segment below is the extended code of gripper connection.

Listing 5.2: Establish connection between base and gripper

```
1   // Initialise interconnect command to current gripper position.
2   comm->command.mutable_interconnect()->mutable_command_id()->set_identifier(0);
3   comm->gripper_command = comm->command.mutable_interconnect()
4       ->mutable_gripper_command()->add_motor_cmd();
```

In function *robif2b_kinova_gen3_update(struct robif2b_kinova_gen3_nbx *b)* , we can add the following line to control the gripper by simply passing the gripper position to *gripper_pos_cmd*

6

Evaluation and results

In this research and development project, three experiments are being conducted with respected to three use cases: (1) Grasp object by sliding motion along surface and (2) Perform writing task and (3)Resting elbow manipulation. First, the set up of the experiments will be introduce. Second,the experiments/evaluation you are performing to analyse the use case will be described. At last, the results of the experiments will be delivered. In this chapter, the experiments and evaluation will be divided according to use cases.

6.1 Setup

The three experiment setups in this research and development project are the same but with different initial position. The tool and robot that is being use anther experiments are listed below:

- Kinova® Gen3 Ultra lightweight robot manipulator is attached to a table
- A Robotiq 2F-85 gripper is attached to the 7th joint of the Kinova arm
- The Kinova arm is connect to a laptop with LAN cable
- A Xbox controller

The direction that discuss in this chapter in according to the tool frame, which is equals to the task frame, robot base frame and the world frame.

6.2 Use case 1 - Grasp object by sliding motion along surface

Experiment design

The objective of this case study is to achieve a successful object grasp by sliding the robot manipulator along a contact surface, serving as a proof-of-concept task to set up the necessary infrastructure for task execution. It is assumed that the pose of the object is already known, and the only item present on the table is the object to be grasped, with no obstacles in the way. The contact surface is also well-defined.

The manipulation process begins with the manipulator approaching the contact surface, such as a table with the target object placed upon it. As the manipulator hovers above the contact surface, it

proceeds to advance toward it until physical contact is established. By actively monitoring the velocity along linear Z axis v_{lin_z} in the world frame. If the absolute value of v_{lin_z} for 10 samples is less than a threshold value. The contact between a surface and the robot manipulator is being established. After contact is established, the manipulator slides along the linear x direction for 10 cm $d_x = 0.1$ until it reaches a grasping region. Finally, the end-effector performs a grasping motion.

In the evaluation, first, we will assess the power consumption of all the joints during motion in both contact and contactless scenarios. The goal is to determine whether performing the task on a contact surface leads to an increase in energy efficiency. Since the voltage and current of each joint are being logged during each trial, by using $P_i = V_i A_i$, where P_i is the power consumption, V_i is the voltage and A_i is the current of a particular joint respectively, we can calculate the power consumption of each joint.

Second, we will analyze the joint torque τ_{Ji} for each joint to determine whether the joint torque decreases when contact is established. Where τ_J indicates joint torque and i is the joint number.

The hypothesis for use case 1 posits that in the contact scenario, energy efficiency will be elevated while joint torque will experience a reduction.

(a) The moment when the contact established. The robot moves from the initial position until contact is established

(b) The moment before the robot move along linear X direction. The robot arm moves along Z direction for 25 cm from the initial position

Figure 6.1: The robot pose before it move forward

Setup

Figure 6.2 shows the initial pose of the robot arm of use case 1. To ensure that the robot consistently begins each trial run from its precise starting position, it is necessary to press the B button on the Xbox

controller. The motion will starts at a fixed starting pose $q = \{6.28318, 0.261895, 3.14159, 4.01417, 0, 0.959856, 0.57079\}$ in radian.

Figure 6.2: The initial pose of the robot arm of use case 1

Evaluation

Table 6.1 presents the average voltage, current, and power consumption for each joint in both contact and contactless scenarios. Interestingly, the average voltage remains nearly unchanged across both scenarios. However, a notable observation arises when comparing the average current and power consumption among the joints.

Specifically, joints 1, 3, and 5 exhibit a substantial requirement for power and current. This phenomenon can be attributed to the robot's motion, which involves movement along the linear z-direction while maintaining a fixed z-position to prevent the collapse of the robot arm. When comparing the average current and power consumption between the two scenarios for individual joints, joint 1 and 5 experience significant increases of 68.2% and 90.6%, respectively. This increase is likely a result of the robot's need to maintain the end effector's position mid-air during movement along the linear x-direction towards the grasping position.

Furthermore, joint 3 stands out as its power consumption in the contact case is 40.5% higher than in the contactless case. This discrepancy may be attributed to the increased power required for this joint to move the end effector along the linear x-direction while overcoming friction with the contact surface.

Examining the average joint torque individually, as shown in Table 6.2, reveals that the torque on joints 1 and 5 increases by 41.2% and 47.5%, respectively, in the contactless case. In contrast, joint 3 experiences a 55.8% decrease in torque under contactless conditions. However, when considering the average torque across all joints, the contact case exhibits an average torque of 1.9857 N, while the contactless case surprisingly records a lower average torque of 0.8256 N.

From the analysis of this data, it becomes evident that performing the task on a contact surface can lead to increased energy efficiency in specific joints, such as the shoulder joint (joint 1) and wrist joint (joint 5). There is also the possibility that the energy efficiency of the elbow joint increases, given the increase in joint torque under contact conditions, likely due to the need to overcome friction with the contact surface. Further studies can explore this behaviour with different contact surfaces featuring varying textures.

	Contact			Contactless		
Joint	Average voltage (V)	Average current (A)	Average power consumption(W)	Average voltage (V)	Average current (A)	Average power consumption(W)
0	23.4669	0.1283	3.0093	23.4727	-0.0416	-0.9776
1	23.3939	-0.7882	-18.4433	23.3541	-1.3291	-31.0272
2	3.1786	0.0626	1.4536	23.1764	0.0104	0.2443
3	23.2972	0.5766	13.4307	23.2945	0.3427	7.9853
4	23.3074	0.1844	4.2983	23.2133	0.0976	2.2659
5	23.2875	0.2210	5.1487	23.2839	0.4215	9.8156
6	23.1337	-0.0084	-0.1950	23.1356	-0.0129	0.2994

Table 6.1: Table of the average voltage, current and power consumption of each joint in contact and contactless case

	Contact	Contactless
Joint	average torque (N)	average torque (N)
0	0.2164	-0.1564
1	9.8474	16.7563
2	0.3025	0.9884
3	-3.1071	-1.3727
4	0.3014	0.3063
5	-1.7027	-2.5120
6	-0.0787	-0.1095

Table 6.2: Table of the average torque of each joint in contact and contactless case

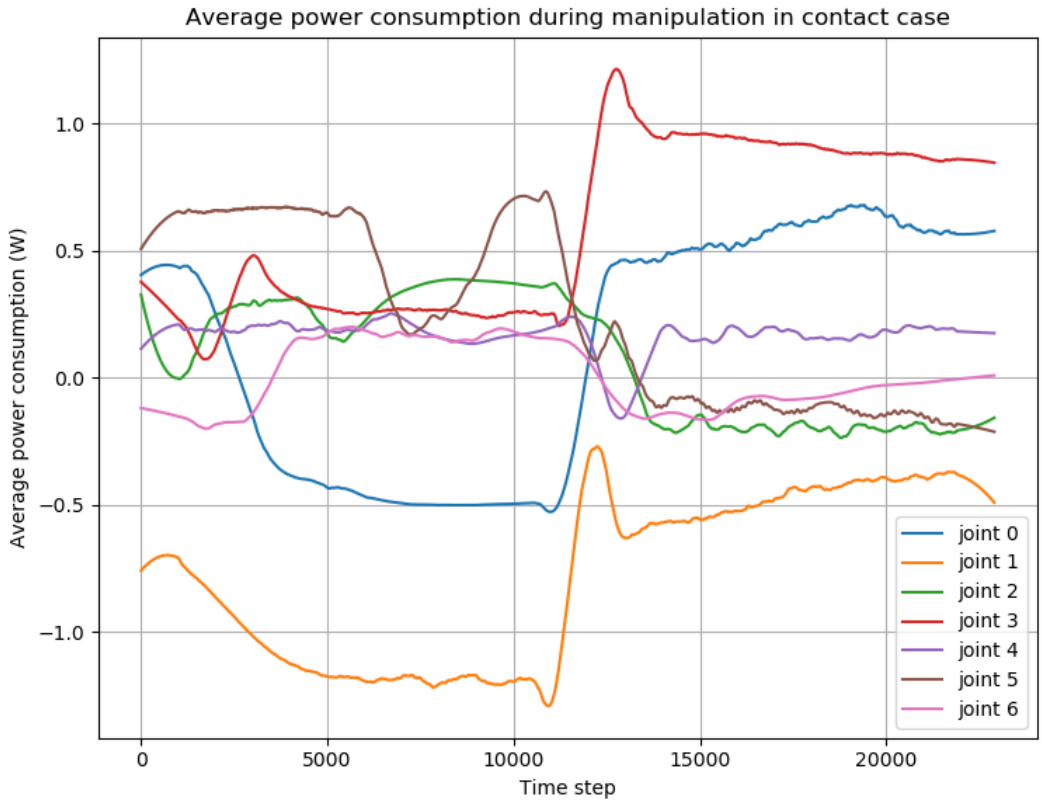


Figure 6.3: Average power consumption during manipulation in contact case

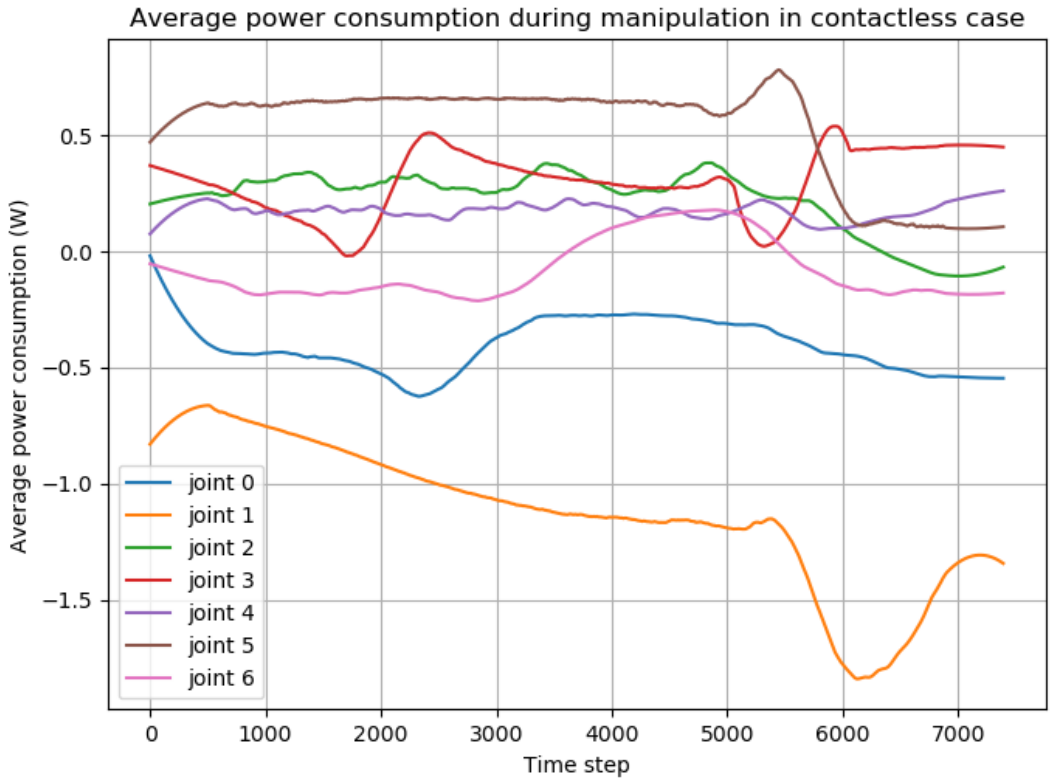


Figure 6.4: Average power consumption during manipulation in contactless case

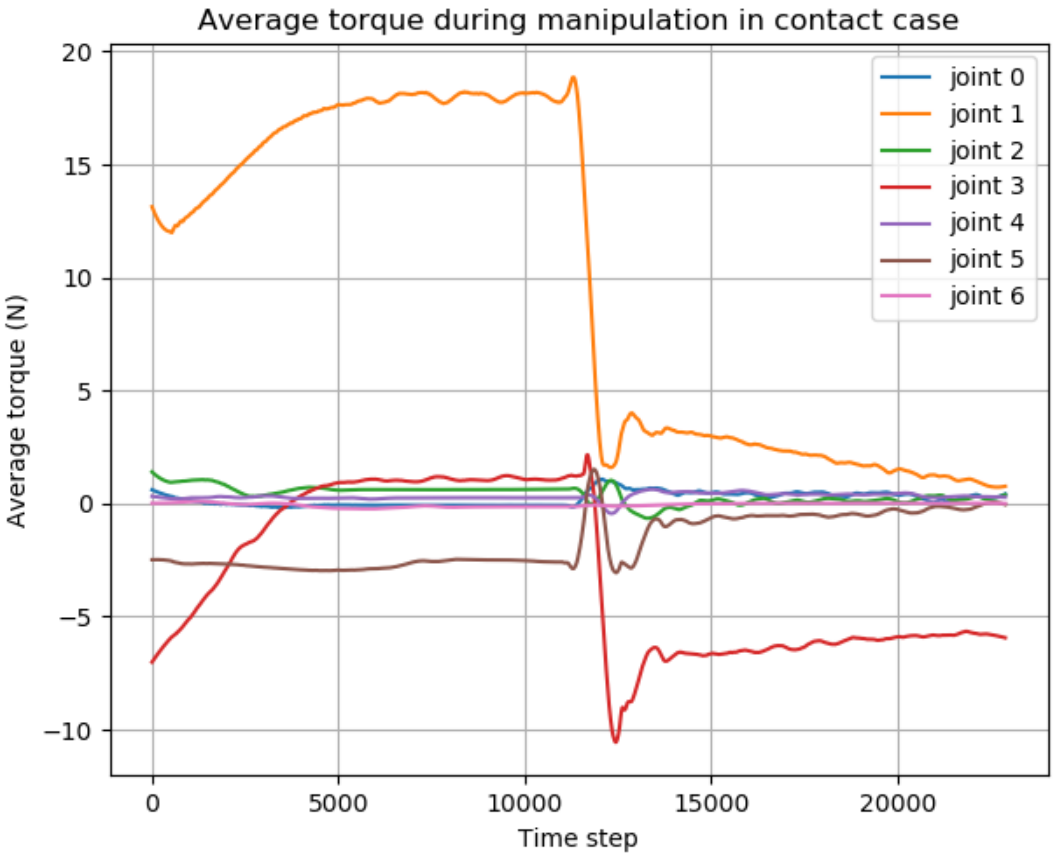


Figure 6.5: Average torque during manipulation in contact case

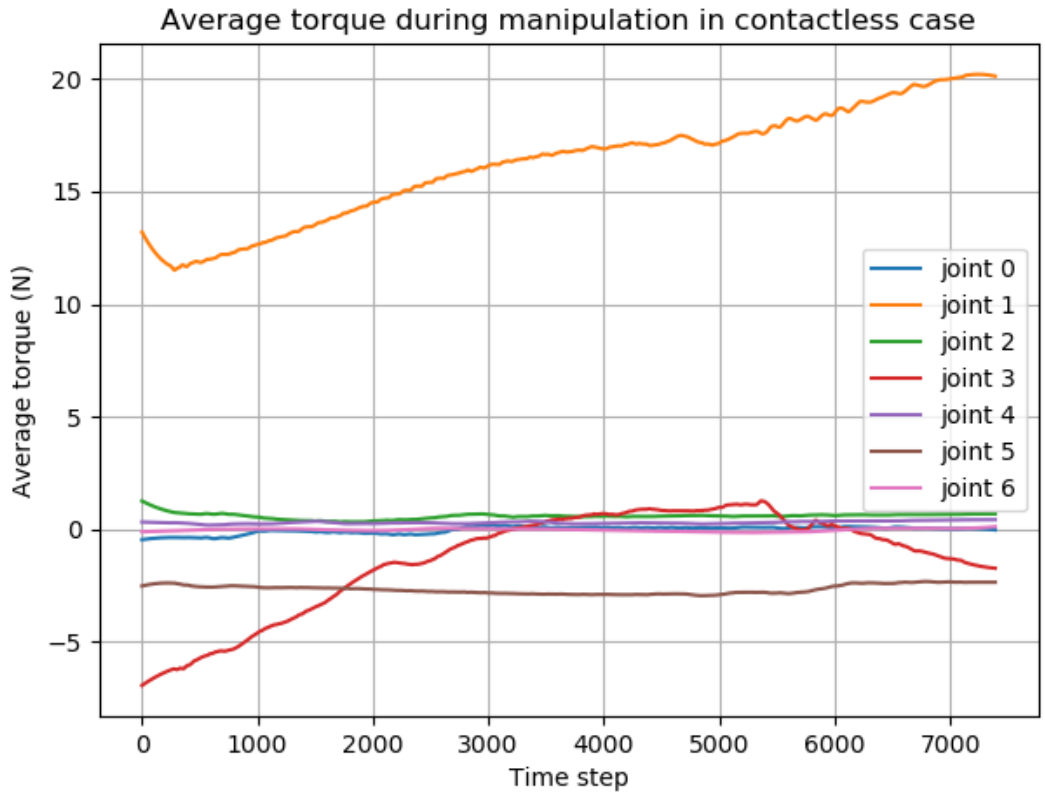


Figure 6.6: Average torque during manipulation in contactless case

6.3 Use case 2 - Perform writing task

Experiment design

The objective of use case 2 is to replicate the act of drawing a line on paper, emulating human writing. To commence the manipulation task, the gripper securely holds the pen or marker. Since this use case focus on the study of if the motion is accurate when robot exploits the contact surface, the motion starts when the marker pen makes contact with the reference starting on as figure 6.7 shows.Upon achieving contact, the robot follows a predefined motion pattern to draw a line which the robot arm moves along linear x direction for 10 cm. The evaluation will involve comparing the trajectory with and without contact between the robot and the supporting surface. We define accuracy as the maximum displacement in linear y direction between a drawn trajectory and the pre-define straight line on the paper.

The hypothesis for use case 2 suggests that in the contact scenario, accuracy will be enhanced, and the trajectory will exhibit greater stability.

Setup

Considering that the purpose of this use case is to perform a writing task, it's important to note that, similar to human writing, the writing motion does not commence from mid-air initially. Instead, the writing task initiates when the tip of the marker pen makes contact with a designated starting reference point on the paper as figure 6.7 demonstrates. In this particular use case, we will conduct two experiments. In the first experiment, we will establish contact before proceeding with the writing task, whereas in the second experiment, no contact will be established beforehand. Figure 6.8a and 6.8b show the initial set up of both experiments at side view. In 6.8b, The ruler indicates that the distance from the centre of the gripper to the table surface is 10 cm. This measurement serves as the initial height for the writing motion in situations where there is no physical contact with the table.

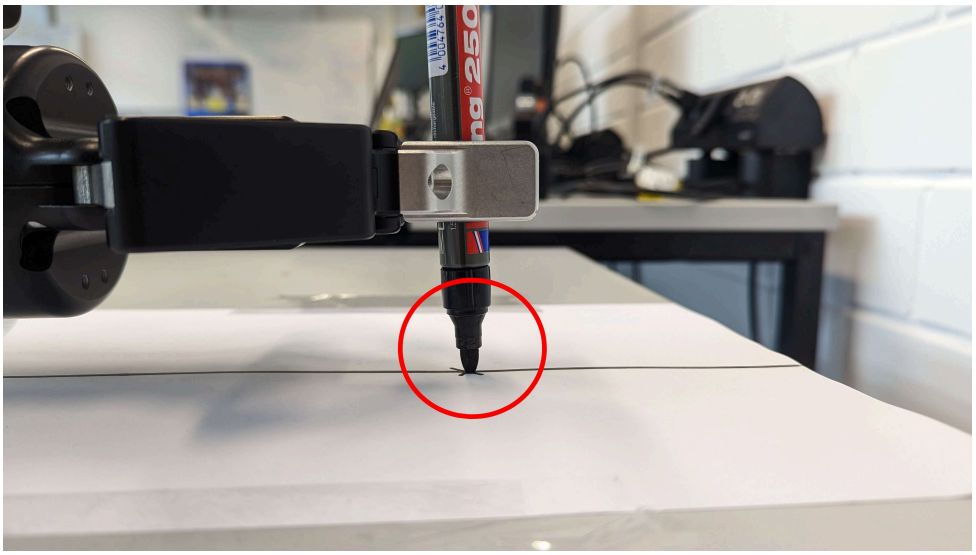
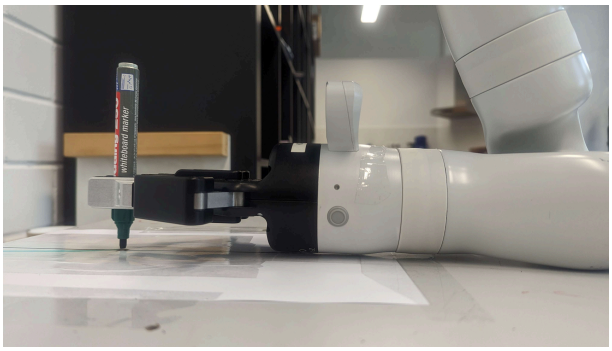
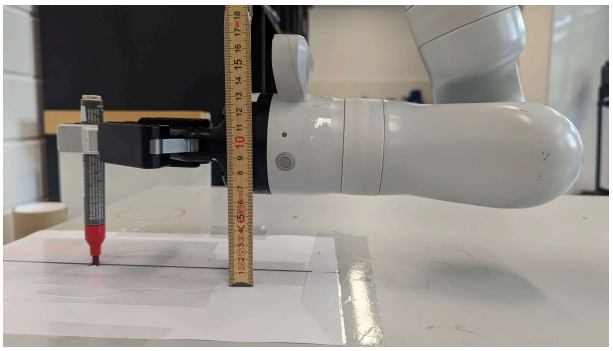


Figure 6.7: Starting position of use case 2



(a) The initial pose of the robot arm of use case 2 in contact condition



(b) The initial pose of the robot arm of use case 2 in contactless condition

Figure 6.8: Initial positions of use case 2

Evaluation

The plots below depict the robot's trajectory with and without contact with the table. In the contact case, as shown in Figure 6.9, the trajectory exhibits misalignment with the reference line at the start of motion, and the displacement between the measured trajectory and the reference trajectory continues to increase. In contrast, Figure 6.10 illustrates the situation in the contactless scenario, where the measured trajectory closely follows the reference trajectory until approximately 0.42 meters, after which it starts to deviate. When we compare both trajectories in Figure 6.11, we observe distinct motion behaviours between the two cases. In the contact case, the motion appears less stable as indicated by the less smooth trajectory, possibly due to friction on the contact surface. Table 6.3 further supports this observation, revealing that the maximum displacements in the contact case are significantly larger than those in the contactless situation, and the absolute average displacements 4 times in the contactless case. This indicates that when the robot arm makes contact with a surface for support in the linear z-direction, considering accuracy based on the average displacement comparison, opting for contactless manipulation of the robot could be viewed as an optimal choice. This observation is unexpected and contradicts the hypothesis of use case 2. This could be attributed to the situation where the constraint on the linear z-direction is removed, leaving the motion subject to the influence of natural forces like gravity. However, the robot must still exert partial control over other directions to ensure that the end effector moves along the x-axis, albeit with greater friction compared to a contactless scenario. During this sliding motion, the end effector might inadvertently shift along the linear x and y axes or even undergo rotation as it attempts to overcome the friction between itself and the contact surface. Given that the surface is not entirely flat, these dynamics may introduce disturbances in various directions of motion.

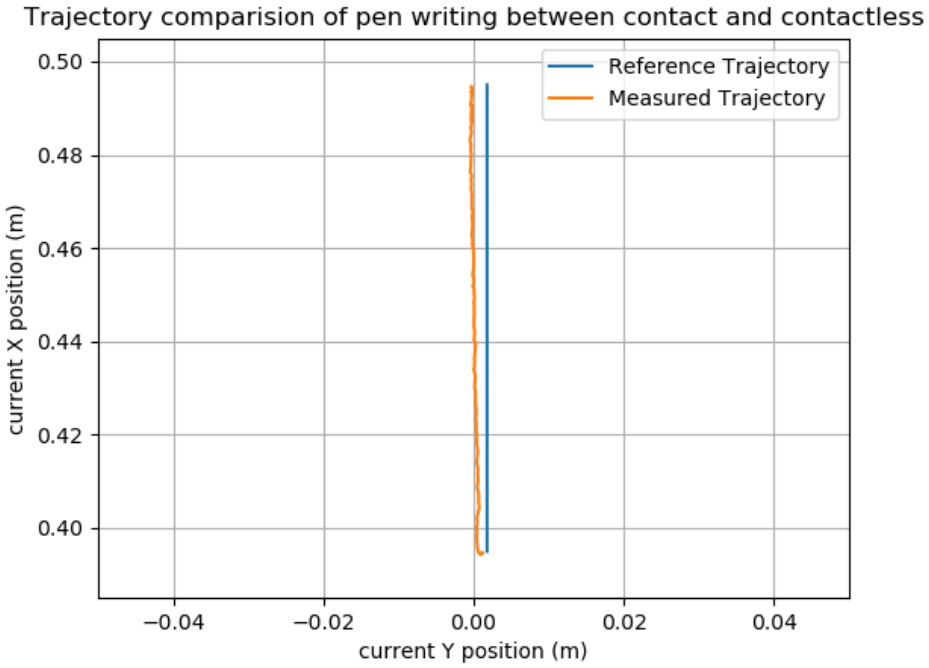


Figure 6.9: The trajectory of pen writing in contact case

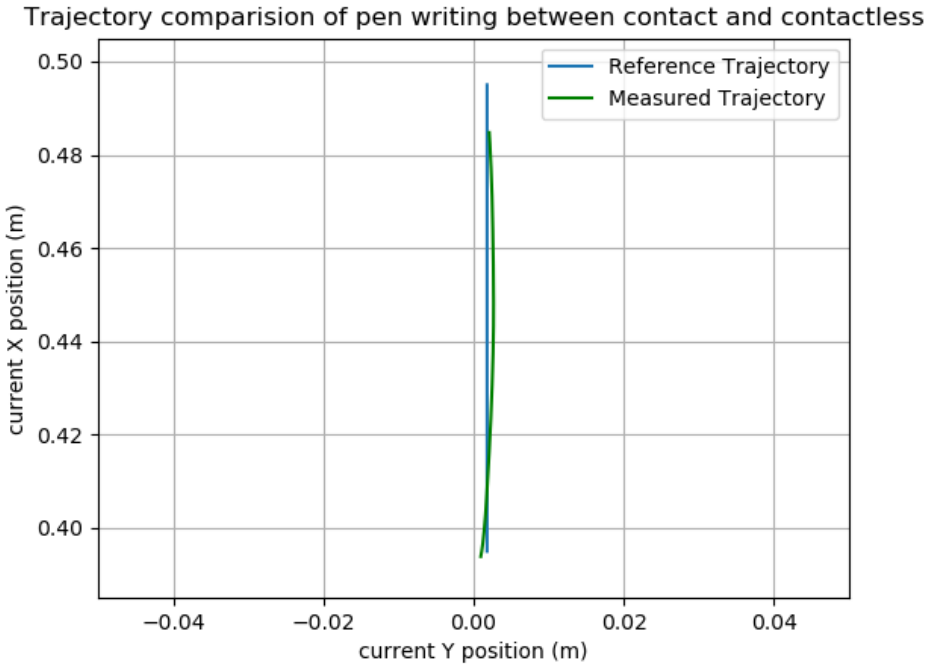


Figure 6.10: The trajectory of pen writing in contactless case

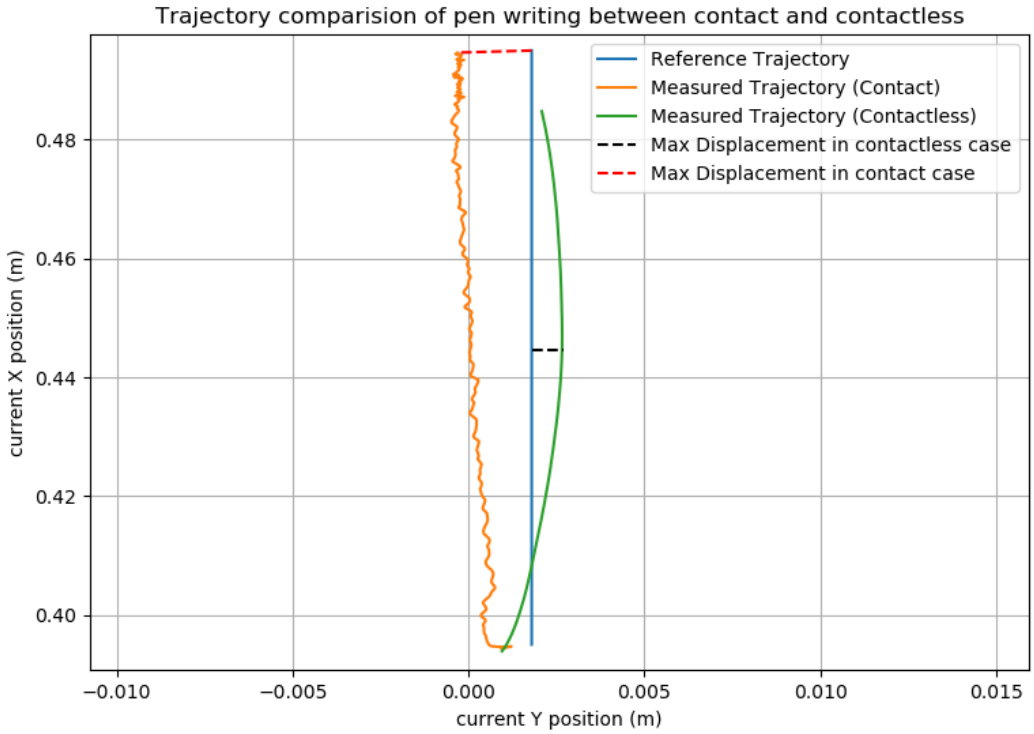


Figure 6.11: Trajectory comparison of writing task between contact and contactless

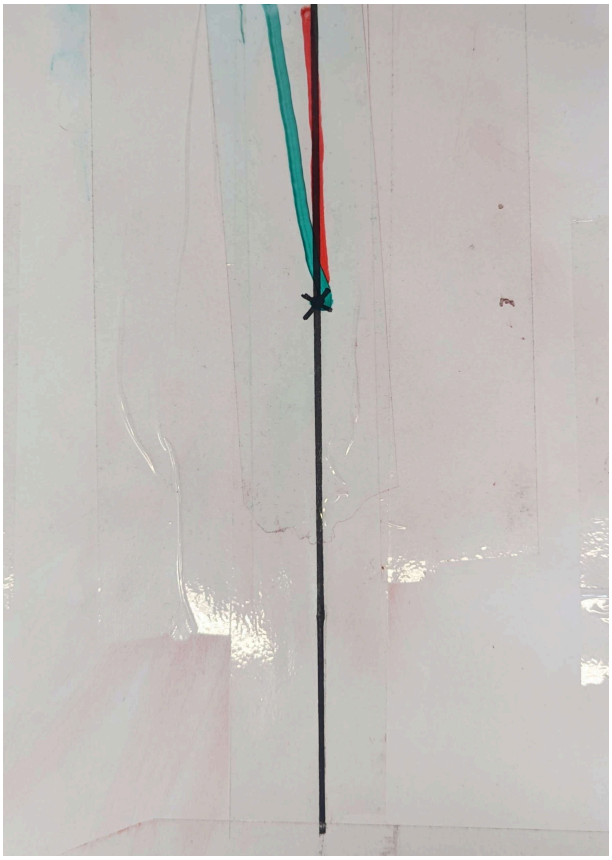


Figure 6.12: The actual trajectory comparison of writing task between contact and contactless task

Picture 6.12 show the actual comparison of the trajectory during experiment.Green line indicates contact case and red line indicates contactless case.

	maximum displacements(m)	average displacements(m)
Contact	0.00211145	-0.00120878
Contactless	0.00089509	0.00034292

Table 6.3: Table of the average displacement, and maximum displacement in contact and contactless case

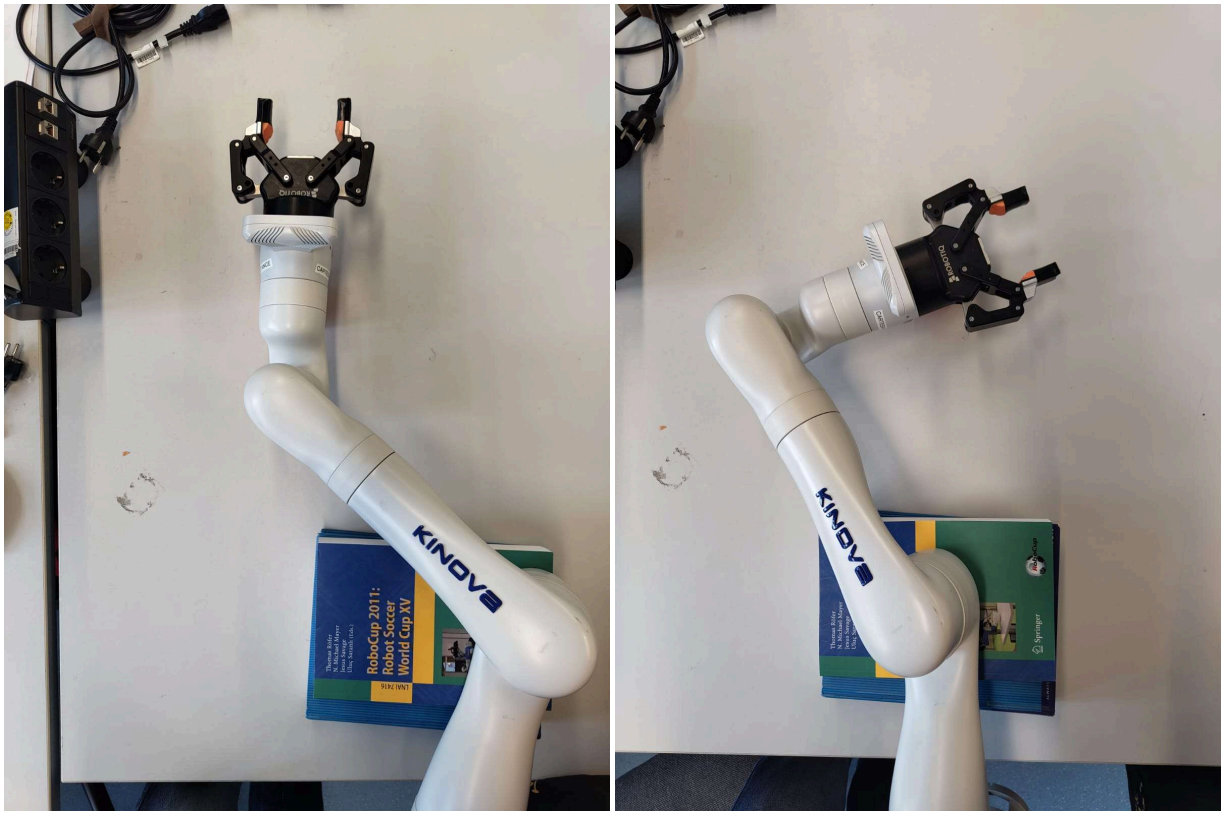
6.4 Use case 3 - Resting elbow manipulation

Experiment design

The objective of this use case is to assess the potential increase in energy efficiency by allowing certain joints of the manipulator to rest on a supporting surface. The manipulator will execute a standardised task, such as grasping object and writing tasks, under two different conditions: one with the joints resting on a surface (e.g., elbows) and another with the joints suspended in midair. The study will involve a comparison of energy consumption during the execution of the same task under these two conditions to determine if resting joints on a surface leads to improved energy efficiency. The hypothesis of use case is that the energy efficiency will be higher as use case 1 demonstrates.

Setup

In this use case, the robot's elbow joint is positioned atop a book, simulating the way humans rest their elbows on a surface when performing tasks such as picking up objects or writing. Figure 6.13a demonstrates the robot set up at a top view.



(a) The initial position of the robot arm in use case 3 where the elbow joint is resting on a book

(b) The final position of the robot arm in use case 3 where the elbow joint is resting on a book

Figure 6.13: Robot arm behaviour in use case 2

Evaluation

A difficult was encountered when conduct the experiment of this use case. As 6.13a showed the initial position. The elbow joint of the robot manipulator is resting on a stack of book and the constraint on linear z direction is disabled but the constraint on the other direction in enabled such that the robot will not rotate and perform grasping task as use case 1. However, when the robot starts the motion, the 6th joint started to rotate in clockwise direction where in the program, the acceleration energy of angular z is being controlled by the cascaded controller where the set point is equals to zero. Means the controller is maintaining zero acceleration energy in angular z direction and joint 6 should not rotate as in figure 6.13b shows. The hypothesis of such behaviour could be the P gains (linear and angular) is not high enough so the end effector cannot rotate back to the set point orientation on time. As the rotational motion led to the 6th joint reaching its limit and subsequently resulted in the termination of the program, the experiment was unable to proceed further. However, it is worth considering that in future attempts, fine-tuning the P gains settings might offer a potential solution for enhancing the motion.

7

Conclusion

In this research and development project, the possibility of exploiting contact constraint in robotic manipulation task has been presented under the experiments with three use cases: grasping object by sliding along a surface, performing writing task and resting robot manipulation.

At the beginning of the report, the state of the art has been presented to introduce the basic concept related to this research and development project. By delivering the concept of dynamics, the related solver: The Articulated Body Algorithm (ABA), Recursive Newton-Euler Algorithm (RNEA) and Articulated-Body Hybrid Dynamics Algorithm (ACHDA) ,we pinpoint out the deficit of the algorithm mentioned above. According to the algorithm delivered in [1] and [2], they all require a full task specification such as all 6 DoF of joint acceleration or joint torque to compute the forward or reverse dynamic which lacks flexibility in real life implementation. Also the concept of controller and some common examples are presented such as Proportional-integral-derivative (PID) controllers, Fuzzy logic algorithm and Impedance controllers. To keep the implementation simple, a cascaded controller with two P-controller is being used. After that the concept of Task based control is delivered to give a brief introduction on how to run a motion by specifying a task instead of directly control individual joints or degree of freedom.

The deficit of state of the art leads to the introduction of Popov-Vereshchagin hybrid solver, and related library being used in this project in chapter 4. The description of the solver, that is the input and the resulting output, have be discussed. Then the task interfaces of Popov-Vereshchagin hybrid solver is being introduced. There are three task interfaces can be specified as input of the solver: The external force F_{ext}, the feed-forward force τ and Cartesian acceleration constraints $\alpha_N^T \ddot{X}_N = \beta_N$. The latest one is the core of the solver where we can assign partial constraint in some degrees of freedom (DOF) instead of a full six DOFs, for example disabling the constraint on linear z direction such that the motion in this direction is defined by nature (gravity). This partial constraint stands out compare to the common dynamic solvers mentioned above. In KDL library, there is an existing function that implemented the Popov-Vereshchagin hybrid solver. In coding, KDL present a overloading function KDL::diff() to calculate the difference between frames, twist .etc which is very practical in implementation stage as we discussed in 4.3.

With the background of Popov-Vereshchagin (PV) hybrid solver and related KDL library are being introduced. Chapter 5 presented the proposed robot architecture of this research and development project. Their system mainly consists of the robot, the PV hybrid solver, and a cascaded controller with the KDL

Parser library to transform the urdf file of the robot manipulator to a KDL chain for dynamic computation. Besides the robot architecture, in software development, instead of using the existing Kinova Kortex API, a robot control interface Robif2b is being used. As we discussed in 5.2, the control interface was lacking connection to the actuator voltage and gripper control. This led to an extension of Robif2b such that we can achieve the function aforementioned.

After discussing the software part of the development, three experiments were conducted according to the use cases. There are three use cases in this project: Grasping objects by sliding motion along the surface, performing writing tasks, and resting elbow manipulation. The experiment designs, setups, and evaluations are presented according to use cases. In use case 1, the general result is expected where the average power consumption of each joint except joint 3 decreases in the contact case, which matches the hypothesis but with an unexpected result where a certain joint has higher energy consumption due to overcoming the friction with the contact surface. In the use case 2, it is surprising that the trajectory of the contact case is less accurate as presented in 6.11 since from a human perspective it is more stable to have our wrist on the contact surface. This may be due to the friction of the surface. The last use case is to study the energy efficiency of resting other joints, for example, an elbow joint on a surface, and perform a manipulation task. According to use case 1, we believe that energy efficiency may increase in such a situation. Additionally, a future investigation can focus on different contact surfaces in terms of texture and shape. With the surprising result in 6.2, it is possible to study how different tasks affect the energy consumption of the robot manipulator because the motion is diver between different tasks and it is possible that other joints would experience a higher torque to overcome the friction just like joint 3 in use case1. From the observation in 6.3 and 6.4, it is possible to study multiple joint resting cases to see if the accuracy increases when the robot task is executed under such a setup.

Future work on the topic of exploiting contact constraints in robotic manipulation tasks could look into fusing force/torque sensor on the robot manipulator since the Kinova arms does not have F/T sensors on the gripper so the task execution is lacking feedback from the gripper to acknowledge the system a gripping task is finished.

References

[1] D. Newton, "Want to be a better writer? try letting a robot tell you what to do," Jun 2017. [Online]. Available: https://qz.com/997006/how-a-robot-improved-my-writing

[2] M. Says, R. K. says, J. says, and P. F. says, "Are paraphrasing tools affecting the development of academic writing skills?" Sep 2022. [Online]. Available: https://www.enago.com/academy/are-paraphrasing-tools-affecting-the-development-of-academic-writing-skills/

[3] A. Shakhimardanov, "Composable robot motion stack," Ph.D. dissertation, 2015.

[4] R. Featherstone, *Rigid Body Dynamics Algorithms*. Berlin, Heidelberg: Springer-Verlag, 2007.

[5] J. Lee, M. X. Grey, S. Ha, T. Kunz, S. Jain, Y. Ye, S. S. Srinivasa, M. Stilman, and C. K. Liu, "DART: Dynamic animation and robotics toolkit," *The Journal of Open Source Software*, vol. 3, no. 22, p. 500, Feb 2018. [Online]. Available: https://doi.org/10.21105/joss.00500

[6] R. Smith, "Open dynamics engine." [Online]. Available: http://www.ode.org/

[7] Admin, "Bullet real-time physics simulation," Mar 2022. [Online]. Available: https://pybullet.org/wordpress/

[8] MuJoCo. [Online]. Available: https://mujoco.org/

[9] R. Featherstone, "The calculation of robot dynamics using articulated-body inertias," *The international journal of robotics research*, vol. 2, no. 1, pp. 13–30, 1983.

[10] ——, "A divide-and-conquer articulated-body algorithm for parallel o (log (n)) calculation of rigid-body dynamics. part 1: Basic algorithm," *The International Journal of Robotics Research*, vol. 18, no. 9, pp. 867–875, 1999.

[11] M. A. Johnson and M. H. Moradi, *PID control*. Springer, 2005.

[12] E. Dadios, *Fuzzy logic: algorithms, techniques and implementations*, 2012.

[13] P. Song, Y. Yu, and X. Zhang, "Impedance control of robots: an overview," in *2017 2nd international conference on cybernetics, robotics and control (CRC)*. IEEE, 2017, pp. 51–55.

[14] R. Smits, T. De Laet, K. Claes, H. Bruyninckx, and J. De Schutter, "itasc: a tool for multi-sensor integration in robot manipulation," in *2008 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*. IEEE, 2008, pp. 426–433.

[15] W. Decré, R. Smits, H. Bruyninckx, and J. De Schutter, "Extending itasc to support inequality constraints and non-instantaneous task specification," in *2009 IEEE International Conference on Robotics and Automation*. IEEE, 2009, pp. 964–971.

[16] D. Vanthienen, S. Robyns, E. Aertbeliën, and J. De Schutter, "Force-sensorless robot force control within the instantaneous task specification and estimation (itasc) framework," in *Benelux Meeting on Systems and Control*, 2013.

[17] N. Mansard, O. Khatib, and A. Kheddar, "A unified approach to integrate unilateral constraints in the stack of tasks," *IEEE Transactions on Robotics*, vol. 25, no. 3, pp. 670–685, 2009.

[18] N. Mansard, O. Stasse, P. Evrard, and A. Kheddar, "A versatile generalized inverted kinematics implementation for collaborative working humanoid robots: The stack of tasks," in *2009 International conference on advanced robotics*. IEEE, 2009, pp. 1–6.

[19] G. Salvietti, M. Malvezzi, G. Gioioso, and D. Prattichizzo, "Modeling compliant grasps exploiting environmental constraints," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 4941–4946.

[20] C. Eppner and O. Brock, "Planning grasp strategies that exploit environmental constraints," in *2015 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2015, pp. 4947–4952.

[21] R. D. Howe and M. R. Cutkosky, "Practical force-motion models for sliding manipulation," *The International Journal of Robotics Research*, vol. 15, no. 6, pp. 557–572, 1996.

[22] G. Kudra and J. Awrejcewicz, "A smooth model of the resultant friction force on a plane contact area," *Journal of Theoretical and Applied Mechanics*, vol. 54, no. 3, pp. 909–919, 2016.

[23] J. Lee, M. Seo, A. Bylard, R. Sun, and L. Sentis, "Real-time model predictive control for industrial manipulators with singularity-tolerant hierarchical task control," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 12 282–12 288.

[24] A. F. Vereshchagin, "Modeling and control of motion of manipulational robots," *Soviet journal of computer and systems sciences*, vol. 27, no. 5, pp. 29–38, 1989.

[25] ——, "Computer simulation of the dynamics of complicated mechanisms of robot manipulators," *Eng. Cybernet.*, vol. 12, pp. 65–70, 1974.

[26] H. Bruyninckx and O. Khatib, "Gauss' principle and the dynamics of redundant and constrained manipulators," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, vol. 3. IEEE, 2000, pp. 2563–2568.

[27] C. F. Gauß, "Über ein neues allgemeines grundgesetz der mechanik." 1829.

[28] E. Ramm, "Principles of least action and of least constraint," *GAMM-Mitteilungen*, vol. 34, no. 2, pp. 164–182, 2011.

[29] S. Redon, A. Kheddar, and S. Coquillart, "Gauss' least constraints principle and rigid body simulations," in *Proceedings 2002 IEEE international conference on robotics and automation (Cat. No. 02CH37292)*, vol. 1. IEEE, 2002, pp. 517–522.

[30] E. P. Popov, "Control of robots - manipulators," *Engineering Cybernetics*, vol. 6, pp. 18–28, 1974.

[31] S. Schneider and H. Bruyninckx, "Exploiting linearity in dynamics solvers for the design of composable robotic manipulation architectures," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 7439–7446.

[32] P. Kulkarni, S. Schneider, M. Bennewitz, D. Schulz, and P. Plöger, "Applying the popov-vereshchagin hybrid dynamics solver for teleoperation under instantaneous constraints," in *2019 19th International Conference on Advanced Robotics (ICAR)*. IEEE, 2019, pp. 673–680.

[33] A. Shakhimardanov, "Composable robot motion stack: Implementing constrained hybrid dynamics using semantic models of kinematic chains," 2015.

[34] D. Vukcevic, "Lazy robot control by relaxation of motion and force constraints," Hochschule Bonn-Rhein-Sieg, Tech. Rep., 2020. [Online]. Available: https://doi.org/10.18418/978-3-96043-084-1

[35] "kdl parser github repository." [Online]. Available: https://github.com/ros/kdl_parser

[36] "Ros wiki page of kdl parser." [Online]. Available: http://wiki.ros.org/kdl_parser

[37] "Robotiq 2f-85 and 2f-140 instruction manual." [Online]. Available: https://assets.robotiq.com/website-assets/support_documents/document/2F-85_2F-140_Instruction_Manual_CB-Series_PDF_20190329.pdf

[38] "Kinova kortex api." [Online]. Available: https://github.com/Kinovarobotics/kortex/tree/master/api_cpp

[39] "Kinova ® gen3 ultra lightweight robot user guide." [Online]. Available: https://www.kinovarobotics.com/uploads/User-Guide-Gen3-R07.pdf

[40] "Rosym-project/robif2b: Building blocks for robot interfaces." [Online]. Available: https://github.com/rosym-project/robif2b