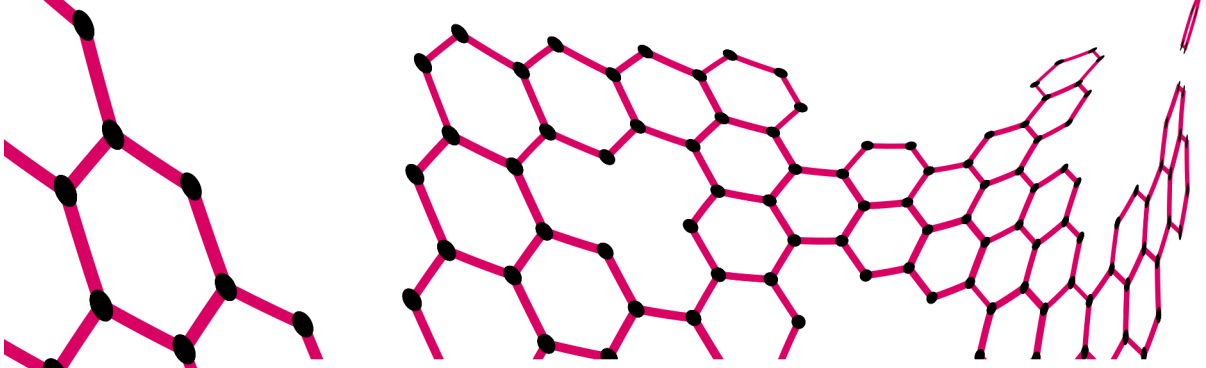

University of Twente

Faculty of Applied Mechanics and Data Analysis

A Comparative Study of Operational Space and Joint Space Control with Null-Space Actuation for a Redundant Manipulator in Obstacle-Rich Environments

Master's Thesis

Author: Gijs Knaken

Supervisor: A. Chatterjee

Date: October 2025

ABSTRACT

Redundant manipulators provide enhanced flexibility through additional degrees of freedom, enabling operation in environments where conventional manipulators would collide with obstacles. By exploiting null-space projection, these systems can avoid obstacles and joint limits while maintaining end-effector tracking. This thesis investigates and compares joint space control and operational space control methods incorporating null-space projection. Operational space control is expected to offer reduced computational cost by eliminating the need for inverse kinematics. In addition, a novel potential field-guided approach with local minima estimation is proposed to further reduce the computational overhead associated with path planning. Three manipulator models were tested in five simulated environments to evaluate tracking performance, obstacle avoidance, and computational efficiency. The results show that operational space controllers achieve comparable tracking performance to joint space controllers with lower computational demand, though at the expense of a higher failure rate. The potential field-guided control method performs effectively in simple environments, but struggles when the goal is obscured by closely spaced obstacles.

CONTENTS

1 INTRODUCTION

1.1 Background

Robotic manipulators are widely used as a solution in all kind of automation tasks where the end effector needs to follow a predefined pose trajectory. These manipulators require at least 6 degrees of freedom (DOF) in order to position its end effector in any 3D pose. When a manipulator has a higher degree of freedom, it is considered redundant. In the case of a redundant manipulator, multiple joint configurations can result in the same end effector pose. These additional joint movements that do not influence the end effector pose are called null space motions. Null space motions can be used to achieve additional tasks next to the end effector tracking task. Examples of these tasks/policies are avoiding obstacles, avoiding joint limits, avoiding singularities, null space damping, and more natural movement [1] [2]

In this report the focus is on the obstacle avoidance (and joint limits) as a secondary task for the null space movement. Obstacle avoidance uses the allowed null space movement to guide the intermediate links of the manipulator away from obstacles. One of the methods of modelling obstacles is by means of artificial potential fields [3][4][5][6]. The intensity of the potential field is directly related to the proximity to the obstacles. The gradient of this potential field provides a repulsive direction away from the obstacles. In addition, the obstacle potential field could possibly be combined with a goal position potential field to navigate the manipulator. In that case, the gradient of the potential field gives the direction in which the end effector of the manipulator is required to travel. A secondary objective of the report is to show the possible use of an additional target position potential field to guide the manipulator. The gradient of this potential field in combination with the environment potential field can be used as a movement vector which moves towards the goal and avoids obstacles. This gradient base steering would eliminate the need for path planning, since the manipulator is guided in real time by the potential field.
Common scenarios in which manipulator redundancy is used to avoid obstacles are welding, inspection, managing warehouses, and agriculture (specifically, harvesting fruits and vegetables). Welding and inspection scenarios require the manipulator to follow a trajectory describing the pose of the end effector of the manipulator. For welding, this pose is of importance to keep a welding torch in the correct orientation, and the redundancy is used to prevent collision of the arm with the working piece. In comparison, warehouse and architecture scenarios do not require a fixed trajectory, and only the final pose is important to get the correct orientation to pick up the plants or to place the warehouse items. The redundancy of the manipulator is used to move around branches in the agriculture case and prevent collision with the shelves in the warehouse case.
The null space actuation and the trajectory tracking of manipulators in these cases are frequently handled by solving an inverse kinematics problem. With inverse kinematics, the trajectory of the end-effector in operational space is translated into a trajectory of joint angles/extensions. The operational space describes the entire reachable space of the manipulator in the real world, whereas the joint space describes the space of all possible joint configurations. The inverse kinematics problem is often solved numerically, taking the obstacles and joint limits into account as constraints to the solver. The resulting joint space trajectories can then be followed by

applying a joint space controller. A big disadvantage of the inverse kinematics implementation is the high computational load. Solving the inverse kinematics problem is an optimization problem which often takes multiple iterations to solve per position time step. Therefore, it may not always be a viable solution on platforms with restricted computing time or available resources. An alternative approach to the trajectory tracking and null space actuation problem is the operational space control based approach [7]. operational space control takes the operational space trajectory directly as input for the control law. The desired null space movements can then be incorporated through null space projection methods [7]. Although this method could be more computationally efficient, it has the disadvantage that the null space projection method does not guarantee obstacle avoidance. This is especially the case when the tracking task has no solutions without obstacle intersections. With inverse kinematics, these situations can be detected before execution in the inverse kinematics process.

1.2 Problem Statement

This report investigates and compares the performance between the inverse kinematics in combination with joint space controller, and the operational space controller with null space projection, in the context of redundant manipulators. Additionally, potential field-based guidance of operational space control is tested as alternative to pre-planned trajectories for point-to-point cases. The manipulator needs to accomplish the tracking with minimal error whilst preventing collisions with multiple obstacles in the operational space. Additionally, to a lesser extent, the manipulator also needs to avoid joint and actuation force limits. The investigation is performed on the above-named welding, inspection, warehouse management, and agriculture cases.
The operational space control method is expected to outperform the common joint space control methods in computational cost, as these controllers do not require inverse kinematics. This advantage gives the control method a better suitability for cases with low computation power available. However, this could impact the performance of the manipulator in other relevant parameters. The joint space controller is expected to outperform the operational space controller in tracking accuracy because the controller only requires tracking of joint angles and not the operational space position. Additionally, the inverse kinematics used for the joint space controller guarantees a joint trajectory without obstacle or joint limit collision. In comparison, the operational space controllers do not have a pre-planned collision-free joint trajectory, and therefore no collision-free trajectory guarantee. The Research objective is to test and compare these performance indicators for each control type.

1.3 Research objective

The objective of this research is to evaluate the relative benefits and limitations of an operational space controller-based approach, compared to a joint space control-based approach for redundant manipulators. In addition a potential field guided operational space control method is evaluated as alternative to the path planning. The study aims to compare these control approaches based on multiple performance criteria: computational efficiency, tracking accuracy, effectiveness in collision avoidance, and the performance of avoiding operation limits of the system.
These performance aspects are analyzed under a range of conditions, including varying trajectories, obstacle configurations, and levels of redundancy of the manipulator.

1.4 Scope

The scope of this project is confined to simulation based testing. A model of a redundant robotic manipulator is evaluated in a scenario containing artificial obstacles, where it is tasked to follow a trajectory generated to avoid obstacles. The scope of the project covers the modeling of the robotic manipulator dynamics and the modeling of the operational space, including obstacles and their corresponding artificial potential fields. In addition, it includes trajectory planning through this operational space. Furthermore, an inverse kinematics solver is to be implemented in combination with a joint space controller. In addition to the joint space controllers, different operational space controllers in combination with their null space controllers are also implemented. One of these control methods is by means of potential-field guidance. Finally, the manipulator model is simulated under the effects of the different controllers and the results are analyzed and documented. All models, control implementations, and simulations are developed using MATLAB.

1.5 Report layout

The report begins with an introduction that explains the background of redundant manipulation and its application in object avoidance. It also outlines the research problem, stating the unknown performance differences between operational space and joint space implementation. Finally, the Introduction gives the scope of the project.
Following the introduction, the theoretical background is given. The theoretical background gives an exploration of the different techniques relevant for the report is given. Following this, the research method is presented. The method explains the reasoning behind the chosen techniques and details how they are used to evaluate the performance of different controllers for the redundant manipulator. The method is followed by a results section. The Results section shows and elaborates on the results from the simulations. The final parts of the report are the discussion and conclusion. These describe the considerations, limitations, and final conclusion of the research.

2 THEORETICAL BACKGROUND

Operational space control of redundant manipulators could be computationally advantageous over classical joint space control. This is due to the joint space actuation that requires inverse kinematics to translate a trajectory in the operational space to a trajectory in joint space. However, operational space control could have several disadvantages over joint space control. These could be trajectory tracking, obstacle avoidance, and joint limit avoidance performance. In this report, the different operational space controllers are compared to the joint space controllers for cases with obstacles in the operational space. In addition, a novel potential field control method with local minima detection is tested, with expected additional computational advantage. In order to execute these tests, prior knowledge is required about key aspects. These key aspects are discussed in this theoretical background chapter of the report.

The following key aspects are discussed for evaluation; the case description, describing the use cases of redundant manipulators and their structure; operational space control, explaining the dynamics equations and the different operational space control methods considered; joint space control, providing different approaches to joint space control; inverse kinematics, explaining different methods transforming an operational space trajectory into a joint space trajectory; null space projection, explaining what null space projection is and the methods of accomplishing it; and finally null space policies, giving an explanation on how to use null space projection to achieve secondary control tasks.
The first paragraph is on the case description and is listed below.

2.1 Case Descriptions

In this case Description paragraph, two major aspects are discussed. The first aspect is the manipulator description. In the manipulator description, an explanation is given on what a redundant manipulator is and what types are considered and discussed throughout the report. The second aspect is the case environment description. In this description, five different scenarios are described in which a redundant manipulator operates and what are some key aspects of these cases.

2.1.1 Manipulator descriptions

A manipulator is described as a series of links connected by one degree of freedom (DOF) joints. These joints can be rotational or translational joints. All possible configurations of these joints are called the joint space of the manipulator. At the end of the manipulator, an end effector is located. An end effector is the functional part of the manipulator that needs to be positioned to perform the tasks. Examples for end effectors are robotic grippers, or welding torches. The degree of freedom of a manipulator describes the number of joints in the manipulator. These degrees of freedom are used to manipulate the end effector pose. The pose of the end effector fully describes its position and orientation in the operational space. This operational space are all possible positions and orientations that the end effector can reach expressed in a Cartesian

world frame. For a planar (2D) manipulator, the number of DOF required to describe the end effector pose is three (two translations and one rotation). For a manipulator in 3D space, the required DOF to describe the end effector pose is six (three translations and three rotations). For this report robotic manipulators in 3 dimensions are considered. If a robotic manipulator has more DOF than the DOF required to describe the end effector position, the manipulator is considered over-actuated or redundant. A redundant manipulator has an infinite number of possible configurations that allow for the same end-effector pose. This is shown for a planar manipulator in Figure 2.1. These additional DOFs can be used for additional tasks, for example, to avoid collision of manipulator links with obstacles.
Redundant manipulators are therefore often used in scenarios where the manipulator operates in-between different obstacles. Several of these environments were identified and evaluated. These are described below in the next paragraph, 'Case environment descriptions'.

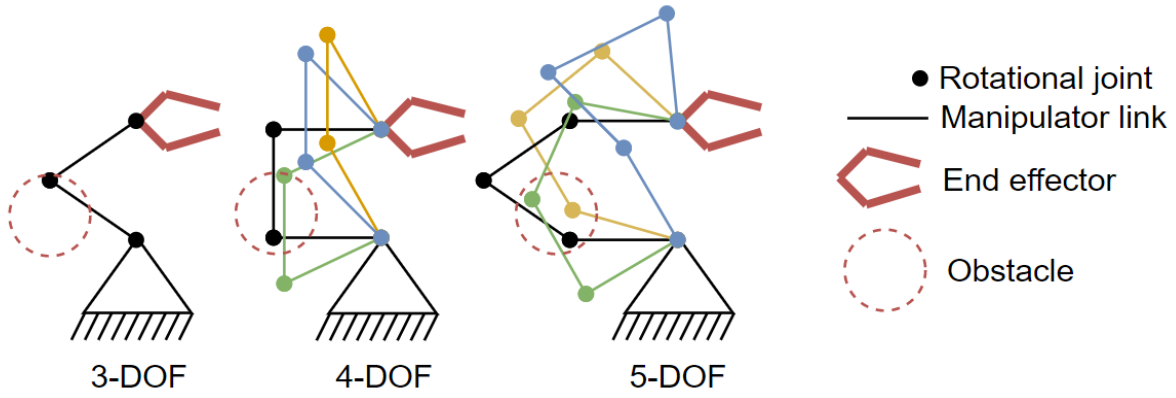

Figure 2.1: schematic showing the redundancy of a planar manipulator

2.1.2 Case environment descriptions

The test cases selected for this study involve applications in which obstacle avoidance is a critical aspect of the task. These cases can be divided into two categories: point-to-point applications and trajectory tracking applications. In point-to-point applications, the robotic manipulator moves from a start pose to an end pose, and the exact path taken is not of primary importance. Three such cases are discussed. The first point-to-point case is a simple obstacle case. This case is for testing the minimum operation capabilities of the redundant manipulator approaches. The second case is warehouse shelf stacking. This case is chosen to represent scenarios with straight trajectories. The last case is the agriculture case, which is chosen to represent a chaotic environment. The second category with cases is the trajectory tracking cases. Two cases of this type are considered: Inspection, in which the manipulator needs to navigate an enclosed environment; and the welding case, in which the manipulator needs to track around an obstacle while keeping its orientation controlled. All cases considered are as follows:

1. **Simple obstacle case** The simple obstacle case is a point-to-point case that represents a minimal case where redundancy is required to move to a end goal without obstacles. In addition, the simple obstacle case can be modelled so that there are local minima in which the manipulator can get stuck. This can help in evaluating the potential field guided operational space control method which is tested in this report. For the other control techniques, this case has a path constructed by a more complex path planning algorithm. as the environment can be chosen to be random and unstructured.

2. **Warehouse case** In warehouse environments, items are often stacked on shelves. When using a robotic manipulator for stacking, it must perform the following tasks: adopt the

correct start and end poses to pick up and place items, navigate between these poses, and avoid collisions with shelves and other products. The last requirement is the main motivation for using a redundant manipulator in this context. The path used in this case can be described by simple line segments. These describe moving in front of the shelve and moving into the shelve. Therefore, this case does not require complex path planning algorithms.

3. **Agriculture case** The Agriculture case represents the picking of fruits and vegetables. To accomplish this, a robotic manipulator must navigate through branches from an initial configuration to a pose for picking. Unlike the warehouse scenario, the orientation of the end effector in agricultural tasks is often less in line with the environment, and the environment is more unstructured, with irregularly placed obstacles. Due to the irregular placement of the obstacles, a complex path planning algorithm is required. that navigates in-between the branches.

4. **Inspection case** The second trajectory tracking case is robotic inspection. Redundant manipulators are used to inspect areas that are inaccessible or hazardous to humans, such as inside machinery or chemical plants. The trajectory is designed to scan regions of interest within the operational space. Redundancy enables flexible navigation in cluttered environments without collisions. This case differs from the welding scenario in both the trajectory shape scanning paths versus weld paths—and the nature of obstacle placement. In inspection tasks, obstacles may be spread throughout the environment, whereas in welding, the tool often operates directly on the obstacle.

5. **Welding case** In this case, the manipulator moves to a starting pose and then follows a trajectory that represents the weld path (e.g., a circular path that joins two pipes). Redundancy helps maintain precise orientation during welding and enables the manipulator to avoid collisions with nearby obstacles, including the objects that are being welded.

The examples for each of the different case environments are shown in Figure 2.2.

In order for the manipulator to operate in these environments, a control law is required that allows the manipulator to follow the trajectories for each of these cases. However, for these controllers, a model description of the manipulator is required. This is discussed in the following paragraph.

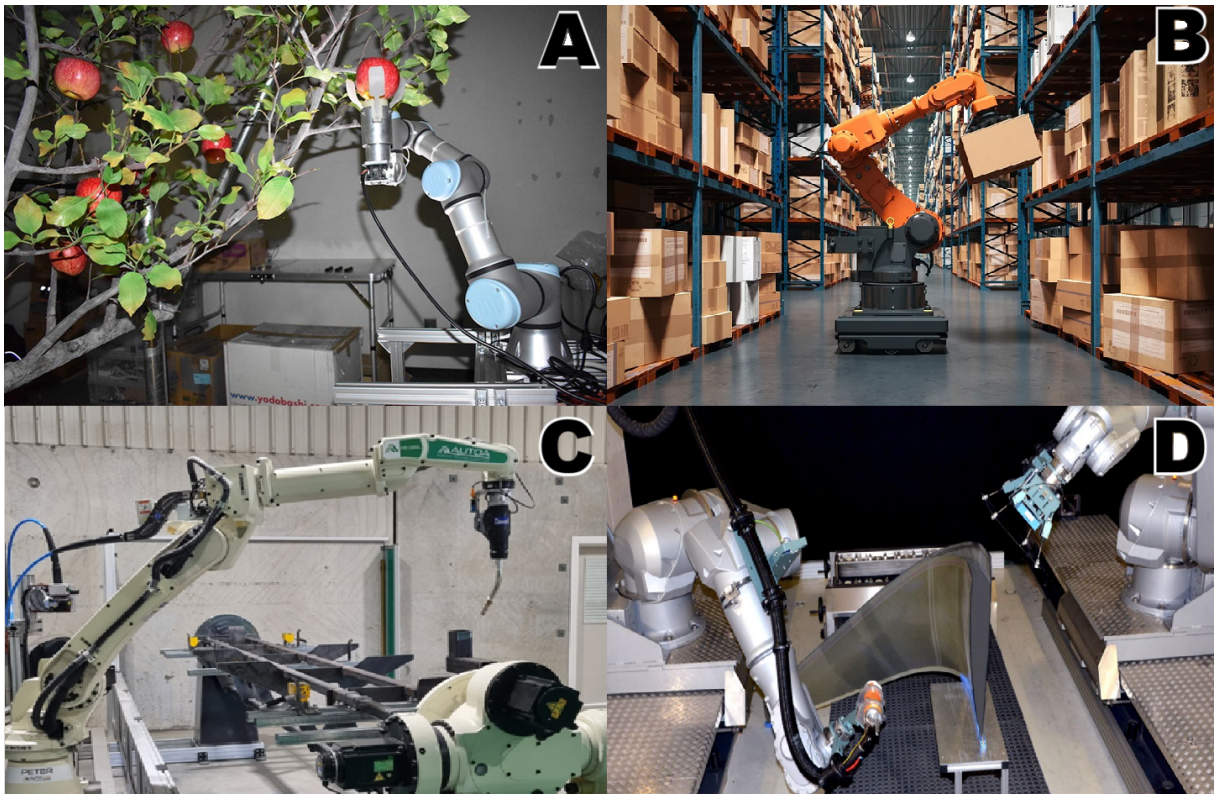

Figure 2.2: Examples of each environment case, (A: agriculture[8], B: warehouse[9], C: Welding[10], D: inspection[11])

2.2 Manipulator model description

As stated before, a manipulator is described as a series of links connected by one-degree-of-freedom joints. In order to move the manipulator, these joints need to be actuated. For the rotational joints, this is done via a torque, for translational/prismatic joints, this is done via a force. The vector with all joint actions is in this report denoted by τ.
By applying a actuation to the joints, The state of the manipulator is changed. This state is a description of the position and velocity of the manipulator. The state is described in the form of each joint position denoted by q and the velocity of each joint denoted by $\dot{q}$. Not part of the state of the manipulator, but important for the dynamics, is the acceleration of the actuators. This is denoted as $\ddot{q}$. Note that the point above q represents the time derivative. The relation between the state of the manipulator, the acceleration and the actuation is given by the dynamics equation as follows:

$$M(q)\ddot{q} + C(q,\dot{q}) + g(q) = \tau \tag{2.1}$$

In this equation $M(q)$ is the mass matrix. This matrix depends on the current configuration of the manipulator and describes the inertia of the system with respect to $\ddot{q}$. $C(q,\dot{q})$ is the Coriolis matrix. The Coriolis matrix describes the effects that the movement of each link has on all other links and depends on the entire state of the manipulator. Finally, $g(q)$ describes the potential energy terms in the system and also depends on the configuration of the manipulator. For the manipulators considered, this would be the actuation's induced by gravity.

The configuration of the manipulator q results in a position of the end effector in the operational space. This position can be calculated by forward kinematics. The kinematics of a manipulator describes the relation of the configuration to the position in the operational space. Positions in the 3D configuration space are described by x,y, and z coordinates, which are for simplicity

denoted as x in this report. Therefore, velocities and accelerations in operational space are denoted by $\dot{x}$ and $\ddot{x}$ respectively.
An additional relation between the operational space and the manipulator state is the Jacobian ($J(q)$). For example, the Jacobian can be used to transform the velocity of the joint space to the velocity of the operation space, and to transform the forces in the operation space (denoted by F) to the forces of the joint space.

$$\dot{x} = J(q)\dot{q} \tag{2.2}$$

$$\tau = J^T(q)F \tag{2.3}$$

In order to use a manipulator in the operation space, a controller is required. The purpose of controllers is to determine an input actuation (τ) so that the manipulator follows a desired trajectory in the operation space denoted by x_d. This x_d is variable over time, as it describes the desired pose of the end effector at a given time point. The tracking of the trajectory is done conventionally using a joint space controller. The operational principle of a joint space controller is described in the following paragraph.

2.3 Joint space controller design

Joint space controllers use desired joint trajectories ($q_d, \dot{q}_d, \ddot{q}_d$) as reverence input for the control law which steers the joint angles. Compared to operational space control, these controllers are generally easier to implement, depending on the desired behaviour and system model. However, the tasks for each of the cases are described in operational space. Therefore, the joint space controllers need a pre-calculated inverse kinematics step which transforms operational space trajectories to joint space trajectories. Additionally, his inverse kinematics scheme handles the object avoidance of the manipulator using the redundancy. The inverse kinematics scheme is the main disadvantage of the joint space controller implementation and the reason why operational space controllers are considered as an alternative in this report. Inverse kinematics is discussed in detail in the later paragraph 'Inverse kinematics'. As the joint space trajectory incorporates all the joints, redundancy does not need to be considered at controller level. A schematic overview of the joint space control schematic is shown in Figure 2.3. Two controllers are discussed here: PID, known for its simplicity and is discussed first, and inverse dynamics for its better tracking performance.

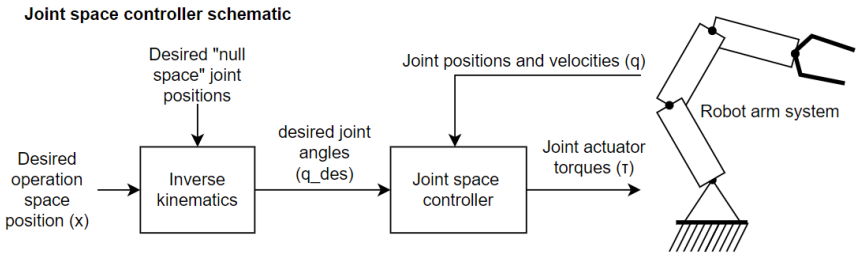

Figure 2.3: Schematic overview of a joint space controller

2.3.1 P(I)D based joint control

The PID controller is widely used in industrial applications due to its simplicity and robustness [12]. It uses proportional (K_p), derivative (K_d), and optionally integral gains (K_i) to minimise the joint tracking error:

$$\begin{aligned} e &= q_d - q \\ \tau &= K_p(e) + K_d(\dot{q}) + K_i \int_0^t e \, dt \end{aligned} \tag{2.4}$$

The controller uses q_d in the differential gain to ensure the stability of the system [13]. The integral term is not strictly necessary to follow a dynamic trajectory and can often be removed. However, in systems with friction or external disturbances, it can help reduce steady-state error. This comes at the cost of potentially introducing instability.
The advantage of the PID controller is that the controller is easy to understand. The P, I and D gains can all be tuned per joint, and controller outputs are predictable for a given state of the manipulator. However, the PID controller has several disadvantages. The main disadvantage of the PID controller is that it does not consider the desired acceleration or system dynamics, which limits its performance in trajectory tracking tasks. Furthermore, it does not account for inertial and Coriolis forces, which can degrade control performance in fast or dynamic motions. Another disadvantage of not considering the dynamics is the difficulty of tuning the PID parameters to achieve desired behaviour. There is a solution to this problem. The problem is solved by using an inverse dynamics controller.

2.3.2 Inverse dynamics controller

The inverse dynamics controller compensates for non-linear dynamics such as Coriolis, centrifugal, and inertial effects by explicitly modelling them in the control law [14]. These dynamics follow from the model described in equation 2.1. The control law is stated as follow and acts as a dynamics compensating pd controller:

$$\tau = M(q)(\ddot{q}_d + K_d(\dot{q}_d - \dot{q}) + K_p(q_d - q)) + C(q, \dot{q}) + g(q) \tag{2.5}$$

The incorporation of the system dynamics in the control law causes the system to be feedback-linearised. Linearisation of the system additionally makes error dynamics linear, resulting in the tracking error converge asymptotically to zero. The errors in tracing are reduced by the error compensating gain K_p and K_d. These gains try to influence the acceleration so that the error in joint configuration and velocity is reduced to 0. The linear nature of the error dynamics makes the adjustment of K_p and K_d relatively easy compared to the PID controller.
The inverse dynamics controller has the additional benefit of incorporating the acceleration of the joint trajectory into the control law. This makes this type of controller more suitable for trajectories with high accelerations.
However, this type of joint space controller also has some disadvantages. The main disadvantage of the inverse dynamics controller is that it is sensitive to modelling inaccuracies. The controller relies on precise cancellation of the non-linear terms, and therefore any error in estimating $M(q)$, $C(q, \dot{q})$, or $g(q)$ can degrade performance or even destabilise the system.

As stated before, both types of joint space controller have the advantage that the controller is directly handled on a given joint space trajectory. Therefore, the controllers themselves do not require the transformation of the operational space trajectory to joint space, making the real-time execution more efficient. However, the transformation is still done offline using a computationally expensive inverse kinematics algorithm. The in this report compared technique of Operational space controllers, improve on this by doing the operation space to joint space transformation on controller level. This removes the requirement of the inverse kinematics entirely. These operational space controllers are discussed next.

2.4 Operational Space Controller Design

Operational space controllers compute actuator torques and forces based on current joint positions, velocities, and accelerations ($q, \dot{q}, \ddot{q}$), as well as the desired operational space trajectory ($x_d, \dot{x}_d, \ddot{x}_d$). This is in contrast to joint space controllers that use joint configurations as a desired

trajectory. The controller's goal is to use τ to act on q so that $x_{endeffector} = x_{trajectory}$. This is schematically depicted in Figure 2.4.

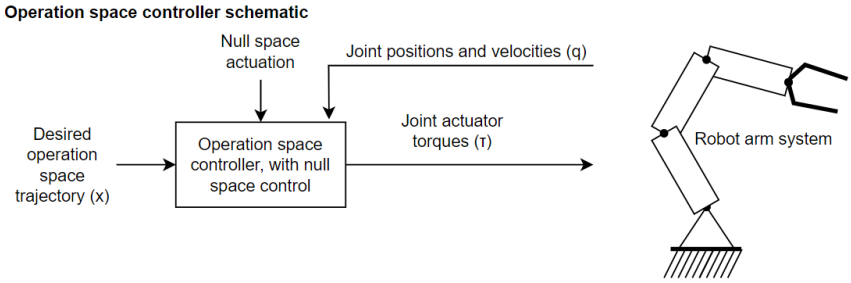

Figure 2.4: Schematic overview of the operation space control implementation

In the operation space control laws, redundancy of the manipulator is also handled by the control law, in contrast to the joint space controller, where it is handled by the inverse kinematics. Redundancy is handled by means of null-space projection. This null-space projection translates secondary tasks as object avoidance and joint limit avoidance to joint movements. The null-space projection is typically integrated as an additional reverence next to $\ddot{q}$ or $\dot{q}$. This means that it wants to change q without influencing $x_{endeffector}$. More details on the working and implementation of null-space control is discussed in the 'Null-space projection methods' and 'Null-space policies' paragraphs. For now, all null-space actuations are denoted by γ_n in all controllers. Here γ_n acts as a secondary joint reverence.
In this report, three different control laws are discussed: Velocity-Based Control, Acceleration-Based Control, and Impedance-Based Control. Additionally, the option for direct null space projection and the incorporation of orientation control are discussed. The first control law evaluated is the velocity-based control.

2.4.1 Velocity-Based Control

Velocity-based control is one of the most robust and simple implementations. The velocity-based control law computes actuator torques and forces using the desired operational space velocity and position, not acceleration[15]. The control law has a compensating term in both the operational space and the joint space, making it robust.
The first step of the velocity-based control law is to translate the measured joint space configuration q into the operational space pose of the end effector. This is done via the forward kinematic function:

$$x = f(q) \tag{2.6}$$

Using this end effector pose, an operational space error compensating control law is derived. This control law uses the error in the desired position $(x_d - x)$ and the desired operational space velocity $\dot{x}_d$ to derive a reverence operational space velocity $(\dot{x}_r$. This reverence control law uses a gain K_p in order to reduce the error, and the formulation is as follows:

$$\dot{x}_r = \dot{x}_d + K_p(x_d - f(q)) \tag{2.7}$$

Next, the reference joint velocity $\dot{q}_r$ is computed from the desired operational space velocity using the Jacobian pseudo-inverse (J^+). The pseudo-inverse is used because the Jacobian is not necessarily square and therefore invertible. At this stage, null space projection (γ_n) is applied to intergrade secondary objectives. This makes the resulting calculation of the reverence joint velocities:

$$\dot{q}_r = J^+ \dot{x}_r + \gamma_n \tag{2.8}$$

For the calculation of the torque are the reverence joint position (q) and acceleration ($\ddot{q}$) also required. These are calculated by differentiation and integration of the reverence joint velocity. In practice, this integration and differentiation is done numerically. However, here they are denoted as exact solutions:

$$q_r = \int_0^t \dot{q}_r \, dt \tag{2.9}$$

$$\ddot{q}_r = \frac{d\dot{q}_r}{dt} \tag{2.10}$$

These quantities are then used to compute the control torque / forces using the system dynamics of equation 2.1. The resulting control law is of the form:

$$\tau = M(q)\ddot{q}_r + C(q, \dot{q}) + g(q) + K_{q,d}(\dot{q}_r - \dot{q}) + K_{q,p}(q_r - q) \tag{2.11}$$

Where $M(q_r)\ddot{q}_r + C(q_r, \dot{q}_r)\dot{q}_r + g(q_r)q_r$ calculates the actuator torques and forces of the dynamics and $K_{q,d}(\dot{q}_r - \dot{q}) + K_{q,p}(q_r - r)$ is a compensating term for position error[7]. The disadvantage of this method is the application of the joint space actuation at the velocity level. In this report, the velocity-based control approach is compared to the acceleration-based control approach, where the null-space actuation is applied as a joint acceleration. Therefore, this method requires different null space actuations. In order to compensate for this and make the comparison fairer. An alternative velocity approach is suggested where the reverence joint velocity is calculated without null-space actuation.

Alternative velocity method

An alternative method to the above-mentioned velocity control method uses the same reverence operational space trajectory but changes the following control laws. First, the reverence joint velocity q_r from equation 2.8, can be calculated without null space actuation as:

$$\dot{q}_r = J^+ \dot{x}_r \tag{2.12}$$

Then $\dot{q}_r$ is only integrated to get q_r (Equation 2.9). The reverence joint acceleration is not calculated by differentiation, as it is calculated as the result of a joint error compensating control law. This control law uses the error of the reverence joint velocity and position with respect to the actual manipulator joint velocity and position. These errors are then multiplied with respective gains K_{dq} and K_{pq} to calculate the reverence acceleration $\ddot{q}_r$. Additionally, the null-space actuation is added resulting in the following formulation:

$$\ddot{q}_r = K_{pq}(q_r - q) + K_{dq}(\dot{q}_r - \dot{q}) + \gamma_n \tag{2.13}$$

Finally, the actuation is calculated using the dynamics of the manipulator using the reverence acceleration as the actuation on the mass matrix. The Coriolis and potential energy matrices use the actual manipulator state in order to compensate for these forces. The resulting formulation for the actuation is as follows:

$$\tau = M(q)\ddot{q}_r + C(q, \dot{q}) + g(q) \tag{2.14}$$

This method may be preferable, since it uses the null-space actuation on the acceleration level. This coincides with the acceleration control discussed hereafter.

The main advantages of velocity-based control are the simplicity in the controller design and the robustness due to stabilisation in the joint space and operational space [7]. Another advantage is the use in the potential-field guided approach as discussed later in the 'potential-field guided operational space control' paragraph. The reverence operational space velocity can directly be replaced by the gradient of the potential field, causing the end effector to find a minima by matching the gradient descent path.
However, there are drawbacks to velocity-based control. one of these drawbacks is the reduced tracking performance due to the absence of explicit acceleration input. This makes the controller less responsive to high acceleration trajectories. [7]. A solution to the lagging tracking performance due to high accelerations is solved by integrating the acceleration in the control law, this is done in the acceleration-based controller.

2.4.2 Acceleration based control

In acceleration-based control, the reference operational space acceleration is computed from the desired trajectory position, velocity, and acceleration[16]. Incorporating acceleration should improve tracking compared to velocity-based approaches. In contrast to the velocity-based controller, the acceleration-based controller uses a reverence acceleration for the operation space instead of a velocity. This reverence acceleration is calculated using the desired acceleration $\ddot{x}$ and two error compensating gains (K_d and K_p) for the desired velocity and position. The resulting reverence acceleration is as follows:

$$\ddot{x}_r = \ddot{x}_d + K_d(\dot{x}_d - \dot{x}) + K_p(x_d - x) \tag{2.15}$$

The corresponding joint accelerations $\ddot{q}_r$ are then obtained using the Jacobian and its time derivative. Additionally, null space control is incorporated at this stage resulting in the following equation:

$$\ddot{q}_r = J^{+}(\ddot{x}_r - \dot{J}\dot{q}) + \gamma_n \tag{2.16}$$

The actuator torques/forces required to achieve these joint accelerations are calculated using the manipulators' dynamic model from Equation 2.1 resulting in the control law:

$$\tau = M(q)\ddot{q}_r + C(q, \dot{q}) + g(q) \tag{2.17}$$

Using the desired acceleration, the controller should theoretically converge the position error to 0. However, in practice, this is impossible because of sampling errors, modelling errors, and other influences. The acceleration-based control law has also drawbacks. A known drawback is that the null space control term is pre-multiplied by the mass matrix. This can cause instability if the mass matrix is not accurately modelled [7]. Additionally, since the null space action operates in the joint acceleration domain, designing a stable and effective null space control law becomes more challenging.
A disadvantage of both the velocity- and acceleration-based controllers is the unpredictable stiffness of the end effector. In most applications, this is not important and. However, in cases where the manipulator needs to interact with a force-sensitive environment, e.g. humans or fruits, this could be of importance. A type of controller that has a predictable stiffness is the Impedance or Force controller.

2.4.3 Impedance/Force Control

The impedance control emulates a spring damper system between the desired and actual end effector trajectory position[17]. Unlike the acceleration-based method, this approach relies only

on the desired position and velocity of the end effector, not on the acceleration. The desired operational space force F_d is computed by modelling a spring damper system. This model is constructed using an error compensating gain (K_p) which acts as the spring constant on the error in position, and a gain on the velocity error (K_d), which acts as a damping constant on the error in velocity. This results in the following desired force equation:

$$F_d = K_p(x_d - x) + K_d(\dot{x}_d - \dot{x}) \tag{2.18}$$

In order to create the desired force in the operation space, the force needs to be transformed to the joint space. This transformation is done via the transpose of the Jacobian as follows:

$$\tau = J^T F_d + \gamma_n \tag{2.19}$$

Traditionally, in this step the null space actuation (γ_n) is added. This null space actuation is a direct force/torque on the actuators.
As stated before, the primary advantage of this type of controller is the predictable stiffness of the manipulator. However, this is not a big consideration for the selected cases to evaluate. An additional advantage of the force controller is the application in the potential-field guided approach similar to the velocity-based approach. The gradient vector of the potential field can be used as a force vector to guide the manipulator to the minimum. More details on the gradient based approach are given in the 'Potential field operational space control' paragraph. The disadvantage of using the force controller in the potential field approach is that the movement is still influenced by the manipulator dynamics, for example, inertia may cause overshoot or even collision with obstacles.
This is also the major drawback of this control law. The lack of any dynamic compensation significantly reduces the tracking performance. Another draw back, similar to the first velocity-based approach, is that the projection is not done on the acceleration level, making a fair comparison difficult. However, this method of projection also shows the possibility to use this projection in other control laws. This will be discussed hereafter.

2.4.4 Direct Null Space Projection at the actuator Level

In the control methods discussed previously, the null space projection (γ) is typically integrated within the control law itself. However, this is not strictly necessary; null space objectives can also be applied directly at the actuator force/torque level [3]. Embedding null space control into the control law has the benefit of satisfying certain null space stability requirements. If stability is not guaranteed in this direct torque approach, it may be compensated by adding damping terms to the null space joints.
This technique of directly projecting null-space actuation is also used in impedance control. The total actuator torques/forces are calculated as follows:

$$\tau_{actuation} = \tau_{controller} + \gamma \tag{2.20}$$

The main advantage of this projection method can be combined with classical operational space control techniques. Another advantage is that this method of null space projection can be applied for all different control types. This would enable the use of the same policies for all implementations, making a fair comparison easy. However, there is a big disadvantage with this projection method. While the controller may consider the manipulator dynamics, the null-space does not. Therefore, the behaviour of the manipulator in the null-space can become more unpredictable.

2.4.5 Orientation control

The control of the orientation of the end effector is not specifically named in all of the operational space control methods stated above. However, it requires some additional attention. All control schemes discussed above use the vector x to represent the pose of the manipulator in the operational space. The position component of x corresponds to the Cartesian coordinates of the end effector, whereas the representation of orientation is more complex. A common method for representing orientation is through quaternions [18]. A unit quaternion α can represent any 3D rotation and is defined as:

$$\alpha = \begin{bmatrix} \eta & \epsilon_1 & \epsilon_2 & \epsilon_3 \end{bmatrix} \tag{2.21}$$

where η is a scalar part and $\epsilon = [\epsilon_1, \epsilon_2, \epsilon_3]$ is a vector part. Any rotation can be represented by an angle φ around a unit axis r. Following this argument, the quaternion components are constructed by:

$$\begin{aligned} \eta &= cos(\tfrac{\varphi}{2}) \\ \epsilon &= r \cdot sin(\tfrac{\varphi}{2}) \end{aligned} \tag{2.22}$$

For orientation control, the error (e) between the desired orientation ($\alpha_d = [\eta_d, \epsilon_d]$) and current orientation ($\alpha = [\eta, \epsilon]$) must be defined. This error is formulated as follows:

$$e_\alpha = \eta_d \epsilon - \eta \epsilon_d + \epsilon_d \times \epsilon \tag{2.23}$$

where $\times$ is the cross product.
For the control schemes, the rotational velocity of the end effector also needs to be defined. This rotational velocity is chosen as ω, representing a vector around which the end effector is rotating, and the magnitude indicates the velocity. With this definition a velocity reverence for an orientation controller can be derived as an example as shown as follows:

$$\dot{\omega}_r = \dot{\omega}_d - K_\alpha e_\alpha \tag{2.24}$$

This is similar to the reverence of the velocity-based controller, as listed before. Combining them would result in the reference pose velocity $\dot{x}_d$, which can be used in the velocity control law.

$$\dot{x}_d = \begin{bmatrix} \dot{p}_r \\ \omega_r \end{bmatrix} = \begin{bmatrix} \dot{p}_d - K_p e_p \\ \omega_d - K_\alpha e_\alpha \end{bmatrix} \tag{2.25}$$

Similar for can be done for acceleration reference:

$$\ddot{x}_r = \begin{bmatrix} \ddot{p}_r \\ \dot{\omega}_r \end{bmatrix} = \begin{bmatrix} \ddot{p}_d - K_d \dot{e_p} - K_p e_p \\ \dot{\omega}_d - K_a \dot{e}_\alpha - K_\alpha e_\alpha \end{bmatrix} \tag{2.26}$$

All operational space controllers; listed above act onto the end effectors position by influencing q. The controllers use a null space projector to accomplish an additional task without influencing the end effector position. The process of this null space projection is discussed below.

2.5 Null space projection methods

Null space projection in the context of robotic manipulator control is the process of projecting an action from the entire configuration space onto the null space of the configuration space. In other words: a desired movement of a redundant manipulator is translated to a closely matching movement that does not impact the end-effector position. This can be used to act on the joints to achieve a certain task such as object avoidance without influencing the positions of the end effectors as shown in Figure 2.1. These additional tasks projected onto the null space are called null-space policies and are discussed in the like-named paragraph. Null space policies are

required in this project as the redundancy of the manipulator is used to move around obstacles and prevent the manipulator to reach joint limits. The null-space projection is implemented in the controllers as shown in the previous paragraph. In these controllers, the null-space actuation is denoted as γ_n.
There are multiple techniques to accomplish this null-space projection. The most common technique uses the Moore-Penrose pseudoinverse, which also forms the basis of the other projection methods.

2.5.1 Pseudo inverse projection

The optimal method of null-space projection is given by the Pseudo inverse[19]. The Moore-Penrose inverse (A^+), or pseudo-inverse, is an inverse that offers an inverse possibility for non-square full-rank matrices such that:

$$\begin{aligned} Ax &= b \\ x &= A^+ b \\ I &= AA^+ \end{aligned} \tag{2.27}$$

In the case of a non-square full rank matrix A, there are an infinite number of matrices for B that satisfy $AB = I$, where I is the identity matrix. However, the pseudo-inverse A^+ is the solution with minimal norm, which means that it is the closest match. The pseudo-inverse can be calculated by means of singular value decomposition or by the following expression:

$$A^+ = A^T(AA^T)^{-1} \tag{2.28}$$

The null space pseudo-inverse projection uses orthogonal projection. Any vector in the operational space (τ) can be split into a vector on the column space of the Jacobian transposed (J^T), this is the part of the operational space vector that acts on the end effector, and an orthogonal vector (v) that is in the null space of J. This null space vector is the part of the operational space vector (τ) that has no effect on the end effector. This projection is realised with the pseudo-inverse as follows:

$$\vec{v} = (I - J^+J)\vec{\tau} \tag{2.29}$$

In order to show that this projector does in fact only act on the null-space, two additional properties of the pseudo-inverse are required:

$$JJ^+J = J \tag{2.30}$$

$$(J^+J)^T = J^+J \tag{2.31}$$

Using these statements, the following steps can be taken to show that any actuation τ projected using this method does not influence the primary task:

$$P = (I - J^+J) \tag{2.32}$$

$$JP = J(I - J^+J) = J - JJ^+J = J - J = 0 \tag{2.33}$$

$$JP\tau = 0 \tag{2.34}$$

This shows that any projected actuation ($P\tau$) has no effect on the end effector as multiplication with the Jacobian does not result in any actuation.
Optionally, the null space projection can be scaled after projection in order to increase its effect. The total projection with a scaling α and the desired action h would be as follows:

$$\gamma_n = \alpha(I - J^+J)h \tag{2.35}$$

This projection can be directly implemented into the control laws.
Other projection methods are possible; however, these would not be the closest match to the desired motion. Nevertheless, there are advantages to these projectors. These projectors are called oblique projectors and are discussed in the following.

2.5.2 Oblique projection

The orthogonal projection is the norm-minimised projection and therefore results in the closest matching manipulator behaviour to the input. However, it is not the only null space projection possible. These other projections are oblique projections and can be used for weighted null space projections. An example of a weighted projection matrix (P) is as follows [20]:

$$P = I - J^T(JW^{-1}J^T)^{-1}W^{-1} \tag{2.36}$$

Here, W can be a positive definite matrix. An example of W would be the diagonal matrix with the weights of each joint. This could be used to avoid the joint limit, where the weight is a function of the angle of the joint, and joints that reach their limit are prioritised. Another example would be object avoidance. If a link located halfway into the manipulator needs to move away of an obstacle, only the joints located between the base and that link can be given a high priority, as these joints only act on the link position in the operational space. Multiple null space projections for different tasks can also be combined. One method is the projection of the priority task.

2.5.3 Task priority projection

One form of combining null space operations is by implementing a task priority method of null space projection. Task priority projection uses null space projectors for lower priority tasks that project onto the null space of the task in a higher priority and in the null space of the primary task. With task priority, it is possible to have a lower priority null space task without influencing a high priority null-space task [21].

In this section some different uses of these different projected tasks are already named, e.g. object avoidance and joint limit avoidance. In the next section, these projected tasks(also named null-space policies) are discussed in greater detail.

2.6 Null space policies

Null-space policies are the secondary objectives that can be incorporated into control laws without interfering with the primary end-effector trajectory. This is done by taking advantage of the redundant degrees of freedom within the manipulator. For this project, the two most significant secondary objectives are the avoidance of obstacles, stabilisation of the null space movement, and the avoidance of joint limits. Other applications that are of lesser relevance for this project include singularity avoidance achieving more human-like motion, or improving tracking performance [1][2]. For obstacle avoidance, a potential field-based approach is preferred as this potential field can also be incorporated in the potential field guided control approach discussed later in paragraph 2.8. The avoidance of obstacles is discussed first in the next paragraph.

2.6.1 Potential field obstacle avoidance

The avoidance of obstacles is a significant policy as it ensures that the redundant manipulators can reach around obstacles without the intermediate links to intersect with them. For each of the cases considered in this report, there are imaginable cases where the redundancy of the manipulator can be leveraged to move around obstacles. As stated before, the potential field approach is the best suited, as it is also used for the potential field guided approach.
Potential fields can be used for obstacle avoidance by generating a virtual potential field function for each obstacle. The potential fields are defined so that there is a high potential for the space intersecting the obstacles and a decreasing potential around it. This results in a downward gradient for the robotic arm to descend in order to move away from obstacles[3][5][4].

The descent from the potential field ($P(x)$) is driven by setting a virtual force (F) at the key points of interest in the manipulator. The virtual forces acting on the points of interest are determined from the gradient of the potential field. Each point of interest is numbered by the variable j. This force can then be transformed into joint torques / force using the Jacobian for the points of interest($J_i(q)$). The formulation of the null-space actuation due to all points of interest can then be formulated as follows:

$$\tau = \sum_{j=1}^{n} J_j^T(q) F(x) \tag{2.37}$$

The potential field function can be derived from the distance between a point and a simple geometric surface[3], or from the distance between two geometric surfaces [22]. An example function for the potential field as a function of the distance between two bodies is as follows:

$$P = \left\{ \begin{array}{cc} \delta \left(\frac{1}{d} - \frac{1}{d_0}\right) \frac{1}{d^2} & d \leq d_0 \\ 0 & d > d_0 \end{array} \right\} \tag{2.38}$$

Where d is the distance between the objects, d_0 is the minimal distance to act, and δ is a scaling factor for the potential field. The scaling acts as a means to tune the speed with which the potential field acts.
The force generated by the potential function can be calculated by computing the gradient of the potential field and multiplying it by -1. The gradient is calculated by taking the partial derivative for each axis as follows:

$$F(x) = -\left[\frac{dP}{dx}, \frac{dP}{dy}, \frac{dP}{dz}\right]^T \tag{2.39}$$

As a null space policy, the calculated actuation can not be implemented directly to the controllers. Depending on the type of projection used and the domain it acts on (joint velocity, acceleration, or actuation), the force needs to be scaled accordingly.
For manipulators with one degree of redundancy (7DOF), an alternative method of obstacle avoidance is available. In this method, a plane is defined within the manipulator arm that can move freely [23]. This plane can then be used as an object of interest that needs to move away from obstacles.

2.6.2 null space damping

Since null-space motions are not inherently stabilised by the primary controller, the system may exhibit an unbounded motion in the null space. To address this, a damping term can be added to the control input to reduce joint velocities in the null space.

$$\tau_{damp} = (I - J^+ J)\alpha\dot{\theta}, \quad \alpha < 0 \tag{2.40}$$

Here, $\dot{\theta}$ represents the joint velocity vector, and α is a negative gain determining the damping strength. This term passively stabilizes null space motion, particularly when no explicit null space objective is defined [24].

2.6.3 Joint limit avoidance

Joint limit avoidance is critical to ensure safe operation and maintain redundancy. When a joint reaches its mechanical limits, it effectively removes one degree of freedom from the manipulator, which can compromise both task execution and safety.
A common strategy is to apply a potential field to each of the joints. The potential increases as the joint angles/extension approach their limits. Similarly to obstacle avoidance, the gradient of

this potential field can then be used to describe the actuation that steers the joints away from the limits [25]. An example of a joint limit potential can be expressed as the following function.

$$P(q_i) = \delta \left(\frac{1}{q_i - q_{min,i}} + \frac{1}{q_{max,i} - q_i} \right) \tag{2.41}$$

The gradient of this potential with respect to q_i gives the torque/force contribution, which can then be projected into the null space of the primary task. This technique is conceptually similar to obstacle avoidance, but is defined in a joint configuration space rather than the operational space.

After all the actuations of the null-space policies are calculated, they can be added to each other and projected using one of the methods discussed in the previous paragraph; or the task priority projection can be used for the projection where the policies are given an order of priority.

2.7 Inverse kinematics

The joint space controllers stated in paragraph 2.3 require the operational space reference trajectory from each of the cases, to be transformed into a joint space trajectory. This transformation process is known as inverse kinematics. The inverse kinematics is computed before the execution of the control loop as shown in Figure 2.3. There are various approaches to solving the inverse kinematics problem, particularly for redundant manipulators where an infinite number of joint configurations can result in the same end-effector pose [26]. Each method handles this redundancy differently. In order to find the most suitable approach, each implementation is discussed here. First, the inverse Jacobian method is considered as it is one of the simplest implementations.

2.7.1 Inverse Jacobian

The inverse Jacobian method for the inverse kinematics problem uses the pseudo-inverse of the Jacobian. This method transforms the desired operational space velocities ($\dot{x}_d$) into joint space velocities ($\dot{q}_d$) using the pseudo-inverse of the Jacobian matrix:

$$\dot{q_d} = J^+(q)\dot{x_d} + \gamma_n \tag{2.42}$$

This method is basically the inverse of the Jacobian transformation of the joint velocity stated in equation 2.2. However, since the Jacobian is non-square for redundant manipulators and there is no singular solution, the pseudo-inverse ($J^+(q)$) is used to give the norm-optimal solution. Additionally, secondary tasks are added here by means of a null-space projection. This null-space projection is required, as it allows obstacle avoidance and other secondary tasks to be implemented without influencing the end effector [6]. Taking advantage of the redundancy of the manipulator.
The joint space velocities can then be integrated and differentiated to obtain the joint positions and accelerations for the controllers. Note, however, that this method relies on perfect matching of the desired path and the joint followed path (which may not be the case due to numerical errors and singularities). This brings forth one of the problems with this inverse kinematics approach. In the case of discrepancy between the desired and realised velocity, e.g. caused by singularities, the integration would result in a drift from the desired position. Furthermore, the null-space projection for obstacle avoidance does not guarantee a solution when the manipulator intersects obstacles. The main advantage of the pseudoinverse method is that the use of the pseudoinverse allows for solutions even in configurations where the system is locally redundant, by returning the closest solution in a least-squares sense.

A solution to the singularity problem of the inverse Jacobian method in addition to an error compensation term is offered by the Damped-lest-squares method.

2.7.2 Damped least squares

The Damped Least Squares (DLS) method [27] is an extension of the inverse Jacobian approach, designed to improve numerical stability near singular configurations. Instead of directly using the Moore-Penrose pseudoinverse, DLS introduces a damping term λ to the inverse operation, which reduces the effect of small singular values that can cause large joint velocities.

$$\Delta q_d = J^T(JJ^T + \lambda^2 I)^{-1} e_d + (I - J(q)^+ J)g \tag{2.43}$$

Here, $e_d = x_d - x$ is the pose error in the operational space and λ is the damping factor. The term $(I - J^+ J)g$ is a null space projector similar to that in the pseudoinverse method, allowing secondary objectives such as avoidance of obstacles or avoidance of joint limits.
Unlike the inverse Jacobian method, DLS computes a position correction Δq_d rather than a velocity, helping to reduce drift over time. However, this means that the method may not strictly adhere to the original trajectory. The main disadvantage of this type of solver, similar to the inverse Jacobian, is that it does not guarantee that obstacles are avoided, especially if the primary tracking task does not allow otherwise. Another disadvantage is that it does not guarantee accuracy in the trajectory.

2.7.3 Optimization based

The optimisation-based approach solves the inverse kinematics problem by iteratively computing the optimal joint velocity $\dot{q}$ that minimises a cost function at each time step [28]. The basic objective is to minimise the following expression, which penalises deviation from the desired end-effector velocity:

$$\min_{\dot{q}} \|J\dot{q} - \dot{x}\|^2 \tag{2.44}$$

This is also done by the pseudoinverse Jacobian method. However, in this case, an additional constraint can be added to the solver. Examples of additional constraints are joint angle limits, actuation limits, and spatial obstacles [29] as shown in:

$$q_{\text{min}} \leq q + \dot{q}\, dt \leq q_{\text{max}} \quad \text{Joint limits} \tag{2.45}$$

$$x = f(q), \quad f(q + \dot{q}\, dt) \neq x_{\text{obstacle}} \quad \text{Obstacle avoidance} \tag{2.46}$$

$$\tau_{\text{min}} \leq M\left(\frac{\dot{q}_{\text{new}} - \dot{q}_{\text{old}}}{dt}\right) + C(q, \dot{q}) + g(q) \leq \tau_{\text{max}} \quad \text{Actuation limits} \tag{2.47}$$

This optimisation can then be solved by any available non-linear multivariable solver, e.g. MATLAB's fmincon function, which uses the SQP optimization algorithm[29]. The disadvantage of this approach is that the solver may deviate from the desired trajectory when finding the optimal solution. A solution is to define the trajectory as a hard constraint and optimise the scaling of the direction error with a scaling value α. where alpha is between 0 and 1, and is tried to maximise.

$$J\dot{q} = \alpha\dot{x} \tag{2.48}$$

This makes the system stop by setting α to zero when no solution exists other than following the trajectory.
In addition to collision detection in the constraints, the potential field acting on the robot (null space) could also be minimised in the optimisation problem [30].
A major advantage of the optimisation-based method is its ability to enforce hard constraints,

such as joint limits, actuation limits, and collision avoidance. Furthermore, with appropriate formulations, it can prioritise either trajectory accuracy or safety by choosing whether to stop or follow an approximate solution when the desired motion is infeasible. The main drawback is its computational cost, as solving constrained non-linear optimisation problems at each control step can be resource intensive. This may limit its use in real-time applications without sufficient computational resources or solver tuning.

The overall big disadvantage of the inverse kinematics implementations is the compute cost. The estimated best approach of optimization-based inverse kinematics takes multiple iterations per trajectory sample and is therefore computationally expensive. The inverse kinematics compute cost can be removed by the operational space controllers, which have their own different disadvantages. The operation space control still requires the path planning. which can also be a large part of the overall computational cost. This is especially the case for a more chaotic environment like in the agriculture or simple obstacle case. In this report, an additional optimisation is proposed for these point-to-point cases using a potential field-guided approach. This is elaborated on in the next section.

2.8 Potential field guided operation space control

The potential field guided operation space control uses the gradient of the potential field to direct the velocity or force that should be generated in the end effector. This should result in the end effector following a gradient descent trajectory towards the goal. The potential field is generated by a goal attractor function in combination with the environment obstacle potential field. The obstacle potential field is easily available since it is already used for the null-space obstacle avoidance. This approach has already been shown to work for 2d cases [31][32]. The challenge of this method is with the formation of local minima. The obstacle field in addition to the goal attractor field could result in local minima where the manipulator guidance would get stuck. One solution is to detect these local minima during operation and place virtual obstacles at these locations [32]. In this report a novel approach for predicting these local minima is tested. This approach uses a first-order approximation of the potential field to estimate the local minima without requiring the computation cost of predicting ahead in the gradient descent.

2.8.1 local minima estimation with first order approximation

The process of finding the local minima in the trajectory of the manipulator can be described in the following steps.
Given the potential field for the goal attractor $V_a(x)$ and the obstacle-potential field $V_o(x)$. A local minimum is defined as a point where the gradient ($g(x)$) of the sum of the potential fields is equal to 0.

$$g(x) = \frac{\delta V_a}{\delta x} + \frac{\delta V_o}{\delta x} = 0 \tag{2.49}$$

Using a first-order Taylor expansion of the system, an estimation can be made for a change in position (δ) where the potential field reaches a equilibrium $g(x + \delta) = 0$

$$g(x + \delta) \approx g(x) + g'(x)\delta = 0 \tag{2.50}$$

$g'(x)$ is the derivative of the gradient and in this 3 dimensional case it is the hessian of the potential field,$H(x)$. Using this information, the equation can be rewritten to solve for δ, which can then be used to find the estimation of the (local) minimum.

$$\delta = -H^{-1}(x)g(x) \tag{2.51}$$

$$x_{min} = x + \delta \tag{2.52}$$

The calculated position x_{min} is a local minimum when the gradient norm is lower than a set threshold ϵ_g. This threshold is required because the estimator is not expected to find the minima exactly.

$$||g(x_{min})|| < \epsilon_g \tag{2.53}$$

For the placement of virtual obstacles, the additional check needs to be executed to see if the minimum found is not the goal position. This is required since the goal position is expected to be the global minima of the potential field. This is done by evaluating whether the minimum is further away from the goal than a set threshold ϵ_e.

$$||x_{min} - x_{goal}|| > \epsilon_e \tag{2.54}$$

3 METHOD

In the Method section, all the methods used to enable testing are elaborated. These methods enable the evaluation between the joint space-space control and operational space control. Furthermore, the novel potential field-guided approach will be evaluated for the relevant cases. The first section is about the modelling of the manipulator. This section discusses the method for modelling the kinematics and dynamics of the manipulator and the choices for the manipulator structure. The second section focusses on the construction of the test cases. This includes the modelling of the obstacles and the accompanying potential field, and the trajectory generation in each case. The third section discusses the implementation of the controllers. This includes the implementation of the null-space projection and policies, and all types of controllers evaluated. The fourth and final section discusses the measurement process used to generate the values used for comparison.

3.1 Manipulator modelling

All testing in this report is done by means of simulation. In order to simulate each controller in all environment cases, models of the manipulators are required. These models describe the dynamics and kinematics of the manipulator for a given actuation and configuration. In addition, the dynamic model of the manipulators is also required for the controller implementation.
In this section. First, the dynamics of the manipulator is discussed. These are the dynamics that describe the state evolution and give the information needed in the controllers. Secondly, the additional dynamics are discussed. These dynamics include properties such as the joint limit behaviour and friction. Third, the different configurations of the manipulator that are used in the test cases are discussed. Different configurations are tested to validate the controllers for different levels of redundancy. Lastly, information is given on the definition of the points of interest. These points are used to interact with the environment. These interactions include collision detection and potential field measurements.

3.1.1 model dynamics and kinematics

To be able to simulate the manipulator, a dynamic model is required. It is chosen to model the manipulator using the Euler-Lagrange method [33]. This modelling method results in a series of coupled differential equations that describe the evolution of the manipulator state. These differential equations can then be simulated using one of the ODE solvers in MATLAB [34], which acts as a variable step integrator. The resulting differential equations can furthermore be rewritten into mass, Coriolis, and potential energy matrices, which are used in the inverse dynamics control laws.
In the Euler-Lagrange modelling technique, the change in potential ($V(q)$) and kinetic ($T(q, \dot{q})$) energy with respect to the generalised coordinates is used to describe the dynamics of the system. Energy is removed or added to the system by toques or forces in the generalised coordinates by τ. In line with the states used in the controllers and for simplicity, the generalised coordinates used for the modelling are chosen to be the actuation of the manipulator joints (q).

These are in radians for the rotational joints and metres for the prismatic joints. Since test cases are evaluated for manipulators with different number of joints, the modelling is kept general for n links. The Euler-Lagrange equation that describes the state evolution in the form of energy exchange is as follows:

$$L = T(q, \dot{q}) - V(q) \tag{3.1}$$

$$\frac{d}{dt}\frac{dL(q,\dot{q})}{dq} - \frac{dL}{dq} = \tau \tag{3.2}$$

The Euler-Lagrange equation requires equations for potential and kinetic energy. These are derived using the twist definition of motion. The kinetic energy is then described as the sum of the energy of each link (i), which are derived using the twist (ξ) and the inertia matrix (I) as follows:

$$T = \sum_{i=1}^{n} T_i = \frac{1}{2}\sum_{i=1}^{n} (\xi_i^{0,0})^T I_i (\xi_i^{0,0}) \tag{3.3}$$

Detailed steps on the derivation of the Euler-Lagrange method, including the kinetic and potential energy derivation, are given in the appendix A. The advantage of using this method is that one of the intermediate steps results in a kinematic model of the manipulator. This kinematic model is used to transform the manipulator state to the end-effector position. furthermore, it is used in order to derive the potential energy function. The only potential energy in the system is due to gravity. This gravity is modelled using the mass of each link (m_i), the position of the centre of mass on the link (P_{mi}), the transformation that describes the pose of the link (H_i), and the gravity vector (g). The potential energy is then calculated as the sum of the potential energies of each link as follows:

$$V = \sum_{i=1}^{n} V_i = \sum_{i=1}^{n} (H_i P_{mi}) g^T m_i \tag{3.4}$$

The equation resulting from the Euler-Lagrange differential equations can be rewritten in a form with a mass, Coriolis, and potential energy matrix, as shown in 2.1. For simulation purposes, the equation is rewritten in state-space form describing the evolution of the manipulator state as follows:

$$\frac{d}{dt}\begin{bmatrix} q \\ \dot{q} \end{bmatrix} = \begin{bmatrix} \dot{q} \\ M^{-1}(q)(\tau - C(q,\dot{q}) - g(q)) \end{bmatrix} \tag{3.5}$$

This state evolution equation describes the rigid body dynamics of the manipulator without any restrictions on the joints. Furthermore, this description only describes the change in energy as τ. However, the change in energy encompasses the input of the actuator, friction, collisions, and other possible external influences. These additional influences are discussed in the next paragraph on additional dynamics.

3.1.2 Additional dynamics

In real life, the dynamics of the manipulators are more complex, friction acts as a non-linear actuation on τ removing energy from the system. Additionally, in real-live the actuators acting on the system have additional dynamic behaviour. Finally, manipulator joints have limits in real life. limiting the operation range. It is chosen to only model the joint limits of the joints and neglect the frictions within the manipulator.
The joint limits are modelled by a collision model. described by the following equations.

$$\tau_{\text{limit}} = \left\{ \begin{array}{cc} K_p \cdot (q - q_{min}) + K_d \cdot \dot{q} \cdot (\cos(\arctan(50 \cdot \dot{q}) + pi) + 1) & q < q_{min}, \dot{q} < 0 \\ K_p \cdot (q - q_{min}) & q < q_{min}, \dot{q} > 0 \\ K_p \cdot (q - q_{max}) + K_d \cdot \dot{q} \cdot (\cos(\arctan(50 \cdot \dot{q}) + pi) + 1) & q > q_{max}, \dot{q} > 0 \\ K_p \cdot (q - q_{max}) & q > q_{max}, \dot{q} < 0 \end{array} \right\} \tag{3.6}$$

Four collision domains are identified. In all domains, a spring force is acting on the joint to pull it out of the collision $K_p \cdot (q - q_{min/max}$. K_p is a spring constant chosen as 1000 to mimic a solid material. For domains where the joint is moving into the limit, an additional damping term ($K_d * \dot{q}$) is added. The additional damping term removes energy from the system when a collision occurs. It is chosen to apply damping only when moving into the joint limit to prevent the manipulator to 'stick' to the limits. In addition, a smoothing factor ($(\cos(\arctan(50 \cdot \dot{q}) + pi) + 1)$) is added on top of the damping to smooth the transition and make it easier to solve when simulating in an ode solver. The complete dynamics of the system including the joint limits can be described by the following state space equation, with τ without subscript, reverencing the actuation applied by the actuators:

$$\frac{d}{dt}\begin{bmatrix} q \\ \dot{q} \end{bmatrix} = \begin{bmatrix} \dot{q} \\ M^{-1}(q)(\tau + \tau_{\text{limit}} - C(q, \dot{q}) - g(q) \end{bmatrix} \tag{3.7}$$

The limits for the joints are chosen as follows. For the x and y joints, the limits are set to $+/-0.75pi$ radians. These limits are chosen to prevent self-intersection of the manipulator. The z-rotation joints have no limit defined, as they can rotate freely without directly causing the intersection between the link segments. The last joints used are the z-translation joints. These joints are limited to operate from 0 to 120 cm. It is chosen to limit these joints for the purpose of realism. The exact configurations chosen for each of these segments are discussed below.

3.1.3 Used Manipulator configurations

Three different configurations of manipulators are chosen. Each configuration is for a different level of redundancy. The three different levels of redundancy considered are 7-DOF, 8-DOF, and 9-DOF. The configurations of each of these models are shown in Figure 3.1. These configurations are chosen in an attempt to enable the most dynamic position movement of the null space. In practice, prismatic joints are not that common. however, in this case they where considered to show the diverse implementation of the control laws.

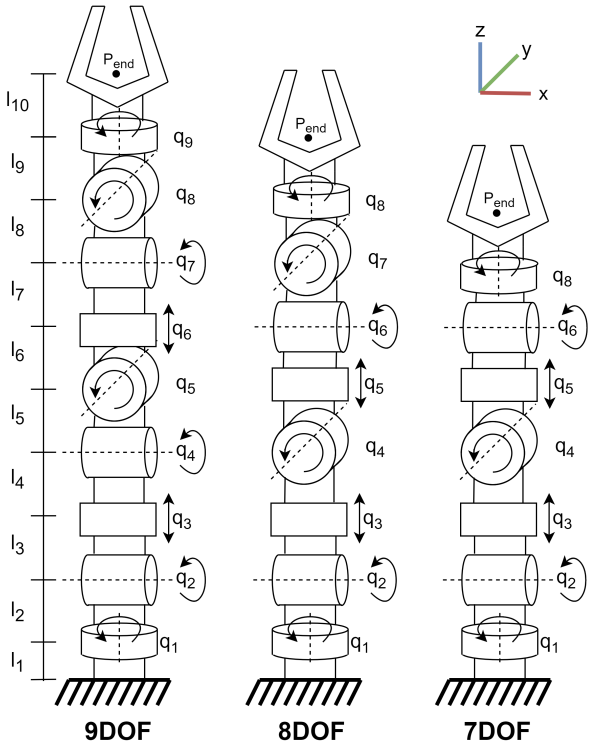

Figure 3.1: Schematic overview of the configurations of the robotic model

For the purpose of getting the inertia matrices for each of the manipulator segments, they are modelled as cylinders with an additional mass on the bottom representing the actuator mass. The total mass of each segment is 2.5 kg. All segments with a rotational joint have a length of 15 cm and all segments of the translation joints have segments of 10 cm. All segments have a width of 10 cm.

3.1.4 Points of interest

In order to evaluate the interaction of the manipulator with the environment and the accompanying potential field, it is chosen to define discrete points of interest. The points are equally spaced over the length of each segment of the manipulator. In this project, four points of interest are placed equally spaced in the centre axis of each manipulator segment. With these points, the distance from obstacles in the environment can be determined. Using this distance information, the intensity of the potential field can be determined at that location. Additionally, the distance is used as a collision detection. where if a point of interest is closer to an obstacle than the width of the manipulator, its considered a collision. The exact method used to determine the distance to the obstacles is discussed in the next section on case construction, where it is used to calculate the potential field intensity.

3.2 Case construction

Since controllers are tested by means of simulation, virtual environments need to be constructed in which manipulator models operate. These virtual environments must mimic the structures encountered in a real-world example. The structure of these environment models are discussed in this section's first paragraph, 'case structure'. The structures of the cases are constructed from multiple configurations of obstacles. The method by which the trajectory in-between these obstacles is planned is discussed in the second paragraph 'Trajectory planning'. How these obstacles are modelled is discussed second in the final paragraph, 'Obstacle and potential field modelling'. This paragraph also discusses how the potential field of a case is constructed for obstacle avoidance, used in the controllers.

3.2.1 case structure

As discussed in the theoretical background section 2.1.2, five different scenarios are considered in this report: Agriculture, Warehouse, Inspection, Welding and Simple obstacle. Each of these cases are tested multiple times. Therefore, it is chosen to generate the cases automatically from a set of rules describing each type of case. A general description of each case is given here.

Agriculture case

In the agriculture case, the manipulator is placed in a fixed starting pose. Surrounding the manipulator, 3 starting positions are randomly chosen, from which rising and expanding tree structures are generated. These three structures represent possible branches that the manipulator is required to avoid. The trajectory that the manipulator has to follow is from the starting position to a randomly placed pose within the branches. Since orientation is only important while grasping, the orientation control is disabled until close to the obstacle. Because the branches form a random and chaotic environment, the trajectory is generated using an RRT algorithm followed by a polynomial fit. Since this case is a point-to-point case, this case is also evaluated for the gradient guided operational space control. An example of this case is shown in Figure

3.2A.

Warehouse case

In the warehouse case, the manipulator is placed in the centre of two parallel sets of shelves. These shelves have two levels and four horizontal segments each. This divides the shelves into 8 compartments. In each of these compartments, a random set of boxes is placed up to a amount of 8. These boxes are placed in an orderly order and stacked to a maximum height of 2. The trajectory of the end effector is defined from the starting position up to a position in the shelves where a box is missing. At this end position, a straight orientation of the end effector is of importance for the virtual placement of a shelve item. Since the environment is highly ordered and predictable, the trajectory can be easily generated by moving in straight lines. The first line positions the end effector in front of the shelf and the position in which it needs to end up. The second line moves the end effector into the shelve. Similarly to the agriculture case, this is a point-to-point trajectory. Therefore, it can also be tested for the gradient-guided operational space controlled scenario. An example of this case is shown in Figure 3.2B.

Inspection case

In the inspection case, the manipulator is placed in front of a confined space which has a random orientation on the x,y plane. Within this space, an obstacle in the form of a cylinder is randomly placed, which requires the manipulator to use its redundancy to avoid it. The trajectory that the manipulator needs to follow is divided into two sections. First, the manipulator moves towards the scanning pattern start. In this phase orientation control is disabled up to a distance of 20 cm from the scanning pattern. The second phase of the trajectory is a scanning pattern at the end of the confined space. During scanning, the manipulator is facing the same direction while following the trajectory. An example of this case is shown in Figure 3.2C.

Welding case

In the welding case, a situation is modelled where a cylinder needs to be welded to a square plate. The plate and cylinder are randomly orientated and positioned close to the manipulator. The trajectory of the manipulator in this case first moves in a straight path towards the welding seem and orients the end effector so that it points into the seem. The second part of the trajectory is the end effector following the seem around the cylinder. For the entire trajectory, orientation control is enabled as it is important for the welding process. An example of the welding case is shown in Figure 3.2D.

Simple obstacle case

The simple obstacle case is a point-to-point case designed to specifically challenge the potential field-guided operational space control method. This is achieved by creating at least one local minima in the trajectory while keeping other disturbances to a minimum. In the simple obstacle case, 4 spherical obstacles are placed between the manipulator starting position and a randomly chosen end position. Three of these obstacles are placed in triangular configuration in front of the manipulator, creating the local minima. The position of this triangle is exactly in-between the starting and goal positions. The trajectory planning in these cases is handled by the same RRT approach as in the agriculture case, due to the random nature of the environment. One of the simple obstacle cases is shown in Figure 3.2 E.

The exact method of trajectory planning for each of the cases is discussed next as it has a significant influence on the compute time of each of the cases.

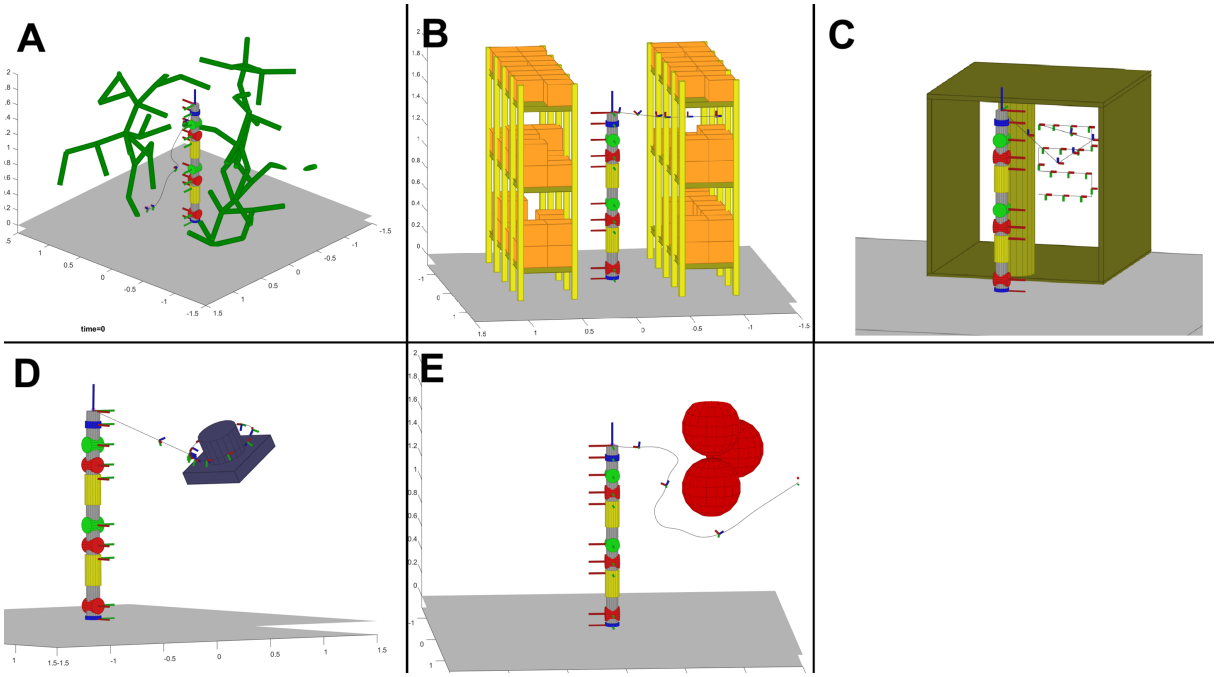

Figure 3.2: Examples of all cases with their respective trajectories plotted and A 9DOF manipulator in the centre.(A:agriculture,B:warehouse,C:inspection,D:welding,E:simple obstacle)

3.2.2 Trajectory planning

Trajectory planning is the process by which a path is generated in the operational space that the manipulator needs to follow. Trajectory planning is a significant part of the computation requirement of the total case execution, and it is desired to remove its need using the potential field-guided approach technique. Therefore, the exact implementation is relevant for discussion. In this paragraph, the two different approaches to trajectory planning are discussed. Geometric trajectory, for the warehouse, welding, and inspection case; and RRT, for the agriculture and the simple obstacle case.

3.2.3 Geometric trajectory

The geometric trajectory is implemented for cases where the environment is known or structured. In these cases, the trajectory is constructed from geometrically described line segments that are designed based on the environment. For the warehouse case, this trajectory is made up of two straight-line segments. One segment moves the trajectory in front of the shelf, and one segment moves the manipulator into the shelf.
The inspection case is also constructed from straight-line segments. In this case the trajectory is planned from the end effector starting location towards a position in front of the confined space where the space between the obstacles in the space and the walls is the biggest. From there a line segment moves straight past the obstacles along the wall. If the obstacles are passed, the trajectory moves in a straight line towards the start of the scanning pattern. This scanning pattern is a 'zig-zag' pattern, mimicking the scanning of a surface.
All straight line segments are sampled so that the trajectory has a smooth velocity curve with at the start and end a velocity of 0. The acceleration and velocity over the line segments are limited to $1m/s$ and $2m/s^2$ to reduce excessive actuation.

For the welding case a straight line segment is used to move towards the cylinder that needs to be welded. The second section of the trajectory is a circle that moves around the cylinder at a fixed distance. The velocity around the cylinder is kept constant with only small sections of acceleration at the start and end.

As the environment is known, no exploration of the operational space for obstacle-free paths is required. This makes trajectory planning computationally efficient compared to the RRT method discussed next. This fact makes these cases less suitable for the potential field-guided control approach, as this approach removes the cost of path planning.

RRT trajectory planning

For cases where the location of obstacles is chaotic with several obstacles, such as in the agriculture case, a more complex algorithm is required. The chosen algorithm is the Recursive Random Tree (RRT) algorithm [35]. This algorithm constructs a point-to-point trajectory that does not intersect with environmental obstacles by randomly exploring a space. It can be implemented to randomly explore the joint space, resulting in a joint trajectory, or to randomly explore the operational space, resulting in an operational space trajectory. Since the comparison of the different control techniques includes the testing of the inverse kinematics, it is chosen to implement the RRT algorithm in the operational space.
The algorithm works as follows. It adds the starting position to a list of branch locations. After that, it randomly chooses a position in the operational space. The closest branch to this position is determined and a step with fixed distance is taken from that branch towards the random position. This step forms a new branch. If this branch intersects an obstacle, a new random position is chosen and the steps are repeated. If the branch does not intersect an obstacle, it is added to the list of branches. This is then repeated until a branch is close enough to the end position.
In this project a adapted version of the RRT algorithm is used (RRT*[36]). This adaptation adds a bias towards the goal position by setting a percentage of the randomly chosen coordinates to the goal position. In addition, a safety radius was implemented around obstacles. This ensures that the manipulator has some room for error around obstacles.
The RRT algorithm results in a set of operational space coordinates that describe the path towards the goal position. However, because the path being randomly chosen, it has large fluctuations that cause high accelerations. To combat this, the path coordinates are filtered with a moving average filter. Additionally, the trajectory needs to be sub-sampled in order to have step sizes matching desired acceleration and velocity limits. This is done by first sampling the path to reduce the number of trajectory samples. Between the remaining samples, 4th-order polynomials are fitted. These polynomials have the conditions that the acceleration and velocity at the end of a polynomial are matched with the start of the next polynomial. All resulting polynomials are then solved for time steps that match the sample time of the controllers implemented later. A flow chart for the entire RRT trajectory planning process is shown in Figure 3.3.
Both types of trajectory planning are implemented discrete in time. However, the controllers that are discussed in the theoretical background and the actual controllers implemented are time continuos. Therefore it was chosen to zero order hold the trajectories at each time sample. This is discussed in more detail in the 'Simulation implementation' section of the controller implementation 3.3.1.

3.2.4 Obstacle and potential field modelling

As shown in 3.2, all obstacles are modelled using simple geometric shapes. These shapes are boxes, cylinders, and spheres. These shapes are chosen because they can be combined to roughly envelop any 3D structure, while keeping it simple to determine the distance to the

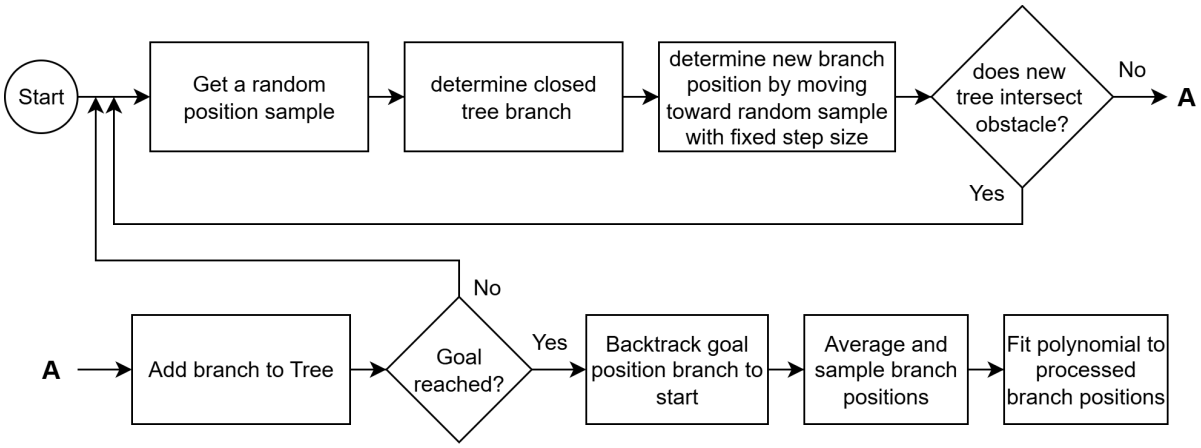

Figure 3.3: Flowchart showing the steps from starting position to trajectory using RRT

object surface anywhere in the operational space. The distance from a point to an obstacle is determined as follows. Each obstacle is placed in the operation space with a transform H. By applying the inverse of the transform H^{-1} to the point of interest, the location of the point is transformed to the coordinate frame of the obstacle. From here, the distance can be calculated by taking the normal distance to the closest surface. If the closest point of the obstacle is an edge or curve, the Euclidean distance is taken. This distance can then be used to calculate the potential and the gradient at that location. However, the calculated gradient must be transformed back into the coordinates of the operating space by applying the transform H. As stated above, the potential field is modelled as a function of the distance to objects. The potential ($V_o(x)$) as a function of the Euclidean distance (x_d) from the surface of an object is defined as:

$$V_o(x) = a(\frac{1}{x_d} + \frac{1}{d})\frac{1}{x_d^2} \tag{3.8}$$

In this equation, a is a scaling factor, and d is the distance in which the potential field is active. These values are chosen according to the use case of the potential field, which is explained in the controller design section. The gradient $g_o(x)$ of the potential field as a function of distance is derived by taking the derivative with respect to each axis. This results in the following gradient expression.

$$g_o(x) = -a \begin{bmatrix} x_d \\ y_d \\ z_d \end{bmatrix} \left(\frac{2}{d x_d^4} + \frac{3}{x_d^5} \right) \tag{3.9}$$

For cases with multiple obstacles, the potential and gradient of all obstacles are added to each other. An example image of the gradient around the warehouse case is shown in Figure 3.4, where red indicates high potential, and blue indicates low potential.

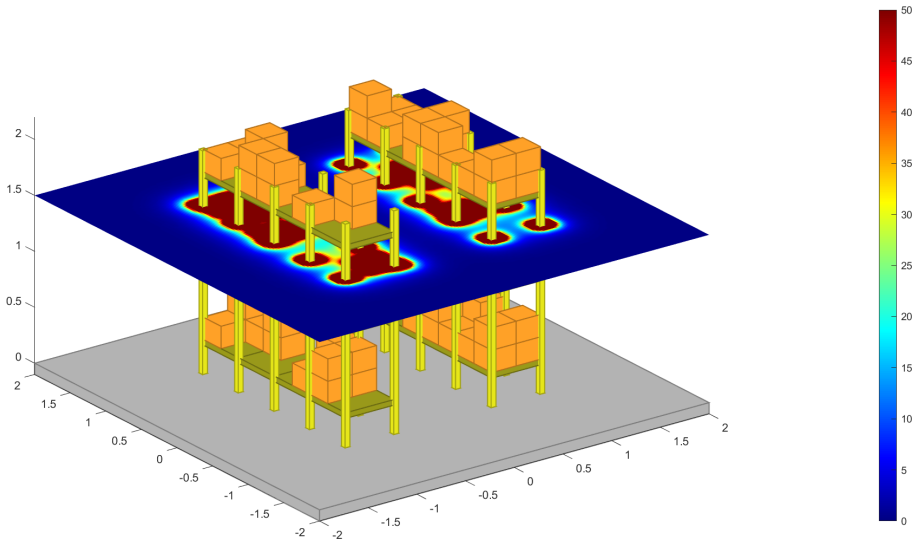

Figure 3.4: A slice of the warehouse case showing the potential field generated by the obstacles

3.3 Controller implementation

From the controller types discussed in the theoretical background, it was chosen to evaluate four different types of controller. Of the operational space controllers, the velocity-based and acceleration-based controllers are tested. Of the joint space controllers it is chosen to evaluate the inverse dynamics controller. For the gradient-guided approach, it is chosen to use an adaptation of the velocity-based controller.
In this section, the simulation implementation of the controllers is discussed first. In the simulation implementation, the realisation of continuous control is explained, as well as how these controllers interact with the time-discrete trajectories generated from the cases. The second paragraph expands on the exact implementation of the control law and the adaptations made from the theoretical background (paragraphs 2.4,2.3). The third paragraph expands on what null-space policies where implemented and how they where implemented. In the final paragraph, the novel implementation of the potential field-guided approach is discussed.

3.3.1 Simulation implementation

The simulation of the manipulator model is performed using an ode solver for each time step. For all types of controller, it was chosen to directly make the control law a part of the state evolution function. This essentially means that each type of controller is implemented continuously. In the real world this would not be possible since controllers and sensors work sample-based and are therefore discrete time. Nevertheless, it was chosen to implement the controllers in real time to ensure the stability of the system without tuning the parameters. Furthermore, this report focusses on comparing the overall control methods. Since both joint space and operational space are implemented equally, it is considered a valid comparison. Note that the trajectory is still sample-based. It is chosen to make the trajectory continuous by applying a zero-order hold which holds the set point fixed for one time sample from $t + ts$. A schematic overview showing the controller implementations is shown in Figure 3.5. For the potential field, the gradient is computed in real time, while the local minima estimate is handled in discrete time. An explanation for this is given in the paragraph 'Gradient guided approach'3.3.3.

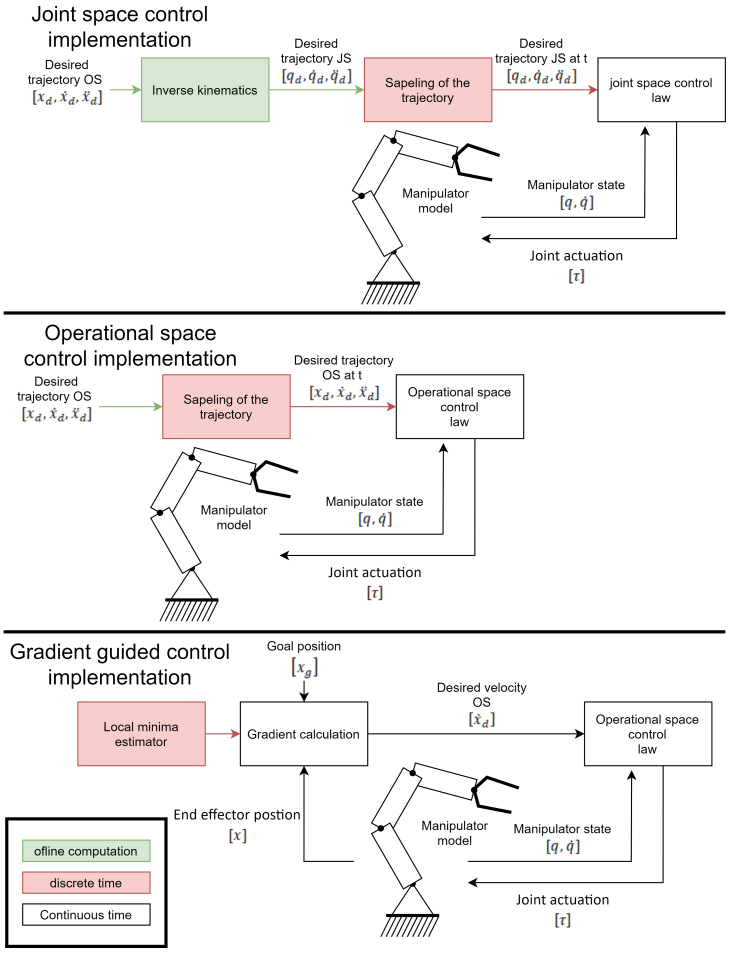

Figure 3.5: Overview of the timing implementation of each section of the control laws

3.3.2 Joint space and operational space based controllers

In this paragraph, the choices for the different control laws and their adaptations are discussed. First, the chosen joint space controller is discussed followed by the two different operational space controllers.

Joint space controller

The joint space controller chosen to evaluate is the inverse dynamics-based approach as discussed in the theoretical background 'inverse dynamics controller'2.3.2. The inverse dynamics control scheme was chosen since it should offer superior performance in comparison to the alternative discussed PID based controller. In addition, since the testing is executed by means of simulation, an exact model is available for the controller design. Therefore, this method of joint space control can be implemented without the risk of instability associated with mismatch in the model used for the controller and the actual system. The control scheme was tuned so that, in combination with the error compensation of the inverse kinematics scheme, the actuator settles at the same time from the input of an impulse for all control techniques. The inverse kinematics required for this controller is discussed in detail in the section 'Inverse kinematics'3.3.6. A comparison between the settling times for all the tracking controllers is shown in Figure 3.6. For the joint space controller, these values were set as $K_d = 100$&$K_p = 1000$.

Velocity-based operational space controller

The velocity-based operational space controller was chosen to test because it is one of the most robust options of the operational-space controllers. From the two discussed velocity-based approaches, the implementation with the reverence velocity and position error used as the desired acceleration is chosen. The description of the control law is given in the theoretical background as the 'alternative velocity method'2.4.1. This choice is made for two reasons. First, as stated in the theoretical background, the alternative method enables the use of the same null-space projector and policies as the acceleration-based controller. Secondly, the alternative method does not require differentiation of the reverence joint velocity. This is a benefit due to the controllers being implemented as part of the manipulator models time-continuous state evolution function. This function is then simulated using an ode solver, which acts as a variable step integrator. The ode-solver can only integrate and not differentiate, making the continuous implementation difficult. The alternative method, in contrast, only uses integration of the reverence joint velocity ($\dot{q}_{ref}$) to get the reverence joint position. This integration is realised by adding the reverence joint velocity as an additional state to the state-space equation used in the ode solver (shown in equation 3.7). The resulting state-space equation including the additional state becomes:

$$\frac{d}{dt}\begin{bmatrix} q_r \\ q \\ \dot{q} \end{bmatrix} = \begin{bmatrix} \dot{q}_r \\ \dot{q} \\ M^{-1}(q)(\tau - \tau_{lim} - C(q,\dot{q}) - g(q)) \end{bmatrix} \tag{3.10}$$

In real-world implementation, both integration and differentiation would have been implemented through a sample-based numeric approximation, which is not possible to use in the variable-time-step ODE solver.
Some changes were made to the velocity controller with respect to the theoretical background. One of these changes is to the error compensation in the reverence joint velocity as follows:

$$\ddot{q}_{\text{ref}} = J^{+}(\dot{x}_d + Ke) + \gamma_n \tag{3.11}$$

This equation uses the term Ke, which is used to denote the operation space rotation- and orientation error multiplied by a gain. This differs from the theoretical background description shown in Equation 2.7, where the position of the operation space and the orientation error multiplied by a gain are simply denoted as $Kp(x_d - x)$. Here, they are described separately, since both use different errors and gains. The error for position stays the same, whereas the rotation error is the quaternion error as described in the paragraph orientation control in the theoretical background.

$$Ke = \begin{bmatrix} K_p(x_{\text{pos,d}} - x) \\ K_{pr}(\eta_d\epsilon - \eta\epsilon_d + \epsilon_d \times \epsilon) \end{bmatrix} \tag{3.12}$$

Another deviation from the velocity-based controller described in the theoretical background chapter is the implementation of the end effector Jacobian J. During implementation a strong difference was observed between the actuation of prismatic joints and rotary joints. This difference was caused by the difference of scale in units used (metres and radians), for the available operation range. This mismatch in scaling was solved by adding a scaling, S_{trans} and S_{rot}, to the translation and rotation parts of the Jacobian. S was chosen to be 1 for translation and 0.2 for rotation gain.
Another adaptation to the Jacobian is implemented due to the choice to switch the orientation control on and off for sections of the desired trajectory. This switching was realised in the control law by adapting the reverence and Jacobian to include or not include the ignore the rotation part. The resulting equations of the Jacobian and reverence are as follows:

$$J = \begin{cases} \begin{bmatrix} S_{\text{trans}} \cdot J_{\text{trans}} \\ S_{\text{rot}} \cdot J_{\text{rot}} \end{bmatrix}, & \text{if do rotation} = 1, \\ S_{\text{trans}} \cdot J_{\text{trans}}, & \text{if do rotation} = 0. \end{cases} \tag{3.13}$$

$$\dot{x}_r = \begin{cases} \begin{bmatrix} \dot{x}_{\text{r trans}} \\ \omega_{\text{r}} \end{bmatrix}, & \text{if do rotation} = 1 \\ \dot{x}_{\text{r trans}}, & \text{if do rotation} = 0 \end{cases} \tag{3.14}$$

Another adaptation that was made was to limit the rotation error. The orientation control loop was found to operate the best with high gain, while having a small orientation error. The orientation error is not expected to be large due to the system having a smooth trajectory from the initial configuration up to the final configuration. However, by implementing a switching orientation control, the orientation error can become relatively high before enabling the controller. To reduce this ”shock” of switching the orientation control, the maximum orientation error (e_{rot}) is limited to a maximum magnitude. In this case, this maximum value was chosen to be a norm quaternion error of 0.075. The error limiting is implemented according to the following equation:

$$e_{rot} = \begin{cases} e_{rot}, & ||e_{rot}|| <= 0.075 \\ 0.075 \frac{e_{rot}}{||e_{rot}||}, & ||e_{rot}|| > 0.075 \end{cases} \tag{3.15}$$

The velocity-based controller has four parameters that require tuning. K_p and K_{pr} are used to compensate for the position and orientation error in the operational space. These values are tuned to 10 and 40. The parameters K_{pq} and K_{dq} are used to compensate for the joint space error and are tuned to 10 and 100. Similarly to the joint space controller, these values are tuned to equal settling time between all controllers as shown in Figure 3.6.

Acceleration base operational space controller

The acceleration based control is chosen as it theoretically should have better performance than velocity-based control in tracking tasks with larger accelerations in the trajectory. Another advantage compared to the velocity-based approach is that no reverence state needs to be differentiated as only the acceleration reverence is implemented. The basis of the implementation is according to the description in the theoretical background paragraph ’Acceleration based control’ 2.4.2.
However, some changes are made to the implementation. One of these changes is the disabling and enabling of the orientation control. Similarly to the velocity-based approach, the switching of the orientation control is handled by enabling/disabling the rotation part of the Jacobian and the reverence operational space trajectory. The reverence also includes different error compensating gains for rotation and position. This results in the following expression for the reverence operational space pose:

$$\ddot{x}_r = \begin{cases} \ddot{x}_d + \begin{bmatrix} K_d(\dot{x}_{\text{d,pos}} - \dot{x}_{\text{pos}}) \\ K_{dr}(\omega_d - \omega) \end{bmatrix} + \begin{bmatrix} K_p(x_{\text{d,pos}-x_{\text{pos}}}) \\ K_{pr}(\eta_d \epsilon - \eta \epsilon_d + \epsilon_d \times \epsilon) \end{bmatrix}, & \text{if do rotation} = 1 \\ \ddot{x}_{\text{d,pos}} + K_d(\dot{x}_{\text{d,pos}} - \dot{x}_{\text{pos}}) + K_p(x_{\text{d,pos}-x_{\text{pos}}}), & \text{if do rotation} = 0 \end{cases} \tag{3.16}$$

The Jacobian and Jacobian derivatives are handled the same as in the velocity control shown in Equation 3.13. This means that for the acceleration control, a scaling factor was also used to promote equal use for the rotation joints and the translation joints. The use of the Jacobian time derivative $\dot{J}$ is unique to the acceleration-based controller. This Jacobian is derived analytically by taking the derivative of the Jacobian with respect to all joint angles individually and multiplying it with the same joints velocity. Each result is summed to achieve the Jacobian time derivative.

$$\dot{J}(q, \dot{q}) = \frac{dJ(q)}{dt} = \sum_{i=1}^{n} \frac{dJ(q)}{dq_i} \cdot \frac{dq_i}{dt} \tag{3.17}$$

Similarly to the velocity-based controller, the acceleration-based controller benefits from a high gain (K_{pr}) for the rotation error. However, like the velocity-based controller, the large gain does

not work with larger rotation errors induced by the switching of the orientation control. Therefore, the same limiting is applied to the orientation error. The rotation error that is limited as described in Equation 3.15, is used in Equation 3.16.
Like the velocity-based controller and the joint space controller, the acceleration-based controller is tuned for an equal settling time, as shown in Figure 3.6. The acceleration based has four parameters that require tuning. K_p and K_{pr} which are the gains to compensate for the error in position and rotation. These values are tuned to be 250 and 2000, respectively. The other two parameters to tune are K_d and K_{dr}, which are gains to compensate for errors in the tracking velocity. The final tuned values are 31 and 100.

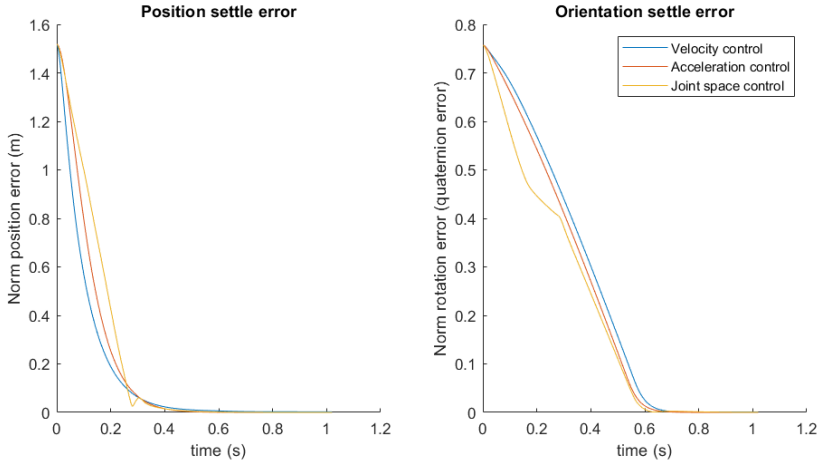

Figure 3.6: This Figure shows a comparison between the settling time from a step input for the different control techniques for a 9-DOF manipulator model

3.3.3 Gradient guided approach

The potential field-guided approach is an alternative to a trajectory-guided approach for point-to-point cases. The benefit of this approach is that it removes the need for path planning, at the cost of requiring a potential field calculation at each time step. The potential field approach works by using the downward gradient as a direction vector for the desired velocity. Since this downwards slope chances magnitude close to saddle points, local minima, and the goal position, the desired velocity magnitude was decoupled from the gradient vector. That is, the gradient $g(x)$ describes the direction of the desired velocity $\dot{x}_d$ and the magnitude is given by a separate function $M(x)$.

$$\dot{x}_d = M(x)\frac{g(x)}{||g(x)||} \tag{3.18}$$

$$M(x) = a(x - x_g)\frac{\tanh\left(\frac{||x-x_g||}{b}\right)}{||x - x_g||} \tag{3.19}$$

$M(x)$ is defined so that a describes the constant velocity magnitude at ”larger” distances, and b describes a slowdown rate close to the goal position g_x. For smaller b the magnitude is reduced slower and if b approaches 0 the function approaches $M(x) = a \cdot \text{sign}\,(x)$. The slowdown was added to prevent oscillations at the goal positions. After testing, these parameters were set to $a = 0.5m/s$ and $b = 0.1$.
The potential function used for the goal attractor V_g was set to an exponential function.

$$V_g(x) = ||(x - x_g)||^2 \tag{3.20}$$

with the accompanying gradient function.

$$g_g(x) = 2(x - x_g) \tag{3.21}$$

This exponential gradient function was chosen because it is exactly matched by the first-order approximation used in the local minima estimator described later.
The potential function of the environment ($V_o(x)$) is used as described in 'Obstacle and potential field modelling' with equation 3.8, and the accompanying gradient ($g_o(x)$) with equation 3.9. This potential function is adapted to have a lower gain of $a = 0.0005$ and a range of $d = 0.5$. This potential function is used because it approaches infinity close to the obstacle. Therefore, the obstacle potential will always be higher than the goal attractor or virtual obstacles.
The total potential field is a summation of the goal attractor, the obstacle potentials, and the virtual obstacles ($V_v(x)$) as follows:

$$V(x) = V_g(x) + \sum V_o(x) + \sum V_v(x) \tag{3.22}$$

$$g(x) = g_g(x) + \sum g_o(x) + \sum g_v(x) \tag{3.23}$$

The virtual obstacle is placed in the operational space by the local minima estimator which is explained below.

Local minima estimator

The local minima estimator is required to prevent the gradient descent from getting stuck in the local minima that can be formed by the interaction of the obstacle and goal potential fields. A slightly modified technique was used from the implementation described in the theoretical background. In the theoretical background, virtual obstacles are placed when a local minima was found and confirmed. However, the final implemented method placed virtual obstacles in all estimated positions. This is done as follows.
At each time step, the location of a virtual obstacle is determined using a first-order approximation described in the paragraph 'local minima estimation with first-order approximation' 2.8.1.

$$x_{min} = x - H^{-1}(x)g(x) \tag{3.24}$$

In this equation H is the Hessian of the total potential field. This Hessian is computed as the sum of the Hessian of the obstacle field, the goal field, and the virtual obstacle field.
The virtual obstacle is then placed at this minimum estimation location. By placing the virtual obstacle directly, its potential field acts on the trajectory faster than when it is only placed exactly at the local minimum. This faster placement results in a smoother and faster trajectory than the technique discussed in the theoretical background.
Initially, it was planned to place only one virtual obstacle at the current estimated local minimum. However, after implementation oscillations were noticed where the potential field of the virtual obstacle placed the next estimation in front of the trajectory. This next virtual obstacle then caused the next estimation to be at the previous location. In order to fix these oscillations, a buffer was implemented where the last 20 local minima estimations are used for the placement of 20 virtual obstacles.
These virtual obstacles have their own potential field description. The chosen potential field function of a virtual obstacle is given by the following equation. This equation describes the potential at x for a virtual obstacle placed at x_v:

$$V_v = \frac{a}{(||x - x_v|| + 1)} - \frac{a}{d + 1} \tag{3.25}$$

With the accompanying gradient:

$$g_v(x) = a\frac{(x - x_v)}{||x - x_v|| \cdot (||x - x_v|| + 1)^2} \tag{3.26}$$

This potential field function was chosen for its reducing gradient and non-infinite value at the placement location. In these functions, d describes the distance in which the potential field acts and is chosen as 0.25 m. If the distance is greater than the acting distance d, the potential is set to 0. a is the amplitude of the potential field, and is chosen to be adaptive depending on the euclidean distance ($||x_{dg}||$) from the goal. This is required because some of the virtual obstacles are placed at the goal position. By making 'a' an adaptive function that reduces the virtual obstacle potential to zero at the goal position. The end effector is able to reach the goal. The chosen function describing a is as follows:

$$a = 0.1 \cdot ||x_{dg}||^2 \tag{3.27}$$

chosen because it scales exponentially like the goal attractor.
A gradient descent trajectory with the local minima estimator implemented is shown in Figure 3.7. This example shows the algorithm working for the simple obstacle case. In this case a local minima is located in front of the red obstacle spheres. As the trajectory approaches this local minimum, the estimation does as well. These local minima estimations generate an artificial potential field that pushes the trajectory around the obstacles.

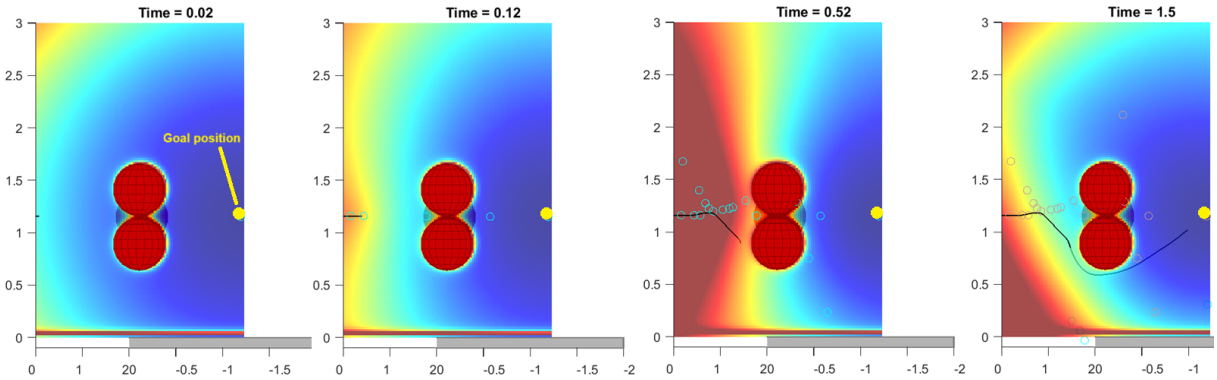

Figure 3.7: Figures showing the gradient descent trajectory (black line) through a potential field, with the cyan dots representing stored active local minima estimations, and gray dots representing previous active local minima estimations

Gradient guided controller

The figure above shows the trajectory if the gradient is directly followed. In order for the manipulator to follow the gradient, a control law is required. The control law used for the potential field guided approach is similar to the acceleration based controller shown in paragraph 3.3.2. However, an adaptation was made to the reverence acceleration $\ddot{x}_r$. The acceleration of reverence is determined only by the velocity error ($\dot{x}_d - \dot{x}$) and not by the position error or the desired acceleration. Furthermore, the orientation control error is only guided by a damping and a goal orientation error compensator.

$$\ddot{x}_r = \begin{cases} \begin{bmatrix} K_d(\dot{x}_{\text{d,pos}} - \dot{x}_{\text{pos}}) \\ -K_{dr}(\omega) \end{bmatrix} + \begin{bmatrix} 0 \\ K_{pr}(\eta_d \epsilon - \eta \epsilon_d + \epsilon_d \times \epsilon) \end{bmatrix}, & \text{if do rotation} = 1 \\ K_d(\dot{x}_{\text{d,pos}} - \dot{x}_{\text{pos}}), & \text{if do rotation} = 0 \end{cases} \tag{3.28}$$

The orientation is not considered for the gradient trajectory as it is only used for point-to-point trajectories. This orientation control is enabled when close to the goal position to give it time to settle to the final orientation. The chosen distance was 0.5 metres.

In the gradient-guided, velocity-based and acceleration-based operational space controllers, null space actuation is denoted by the conventional projection method using the Jacobian pseudoinverse. However, this is a simplification on the actual implementation, which is described in the next paragraph describing the implemented projection methods.

3.3.4 null-space projection and policies

In this paragraph, the projection method of the null-space policies and the null-space policies themselves are discussed.

projection method

The implemented projection method is a variant on the damped pseudo-inverse method. This method uses a damped version of the pseudo-inverse definition (Equation 3.29). The damped pseudo-inverse has the benefit that it reduces the effects of operating close to singular configuration. When the manipulator is close to singular, the values of the undamped pseudo-inverse tend to blow up to infinity. However, damped implementation comes with the disadvantage that it is not an exact inverse (does not result in a unit matrix when multiplied with the non-inverse $JJ^+ \neq I$). The result of this is that the null space projector has an effect on the operational space of the manipulator.

$$J^+ = J^T(JJ^T + \lambda^2 I)^{-1} \tag{3.29}$$

$$\ddot{q}_{\text{r null-space}} = \alpha(I - J^+ J)h \tag{3.30}$$

This problem is minimised by tuning the damping factor λ of the pseudo-inverse. With a smaller λ, the resulting acceleration in the null-space ($\ddot{q}_{\text{r null-space}}$) will have less effect on the operation space x. However, a smaller λ will result in larger accelerations close to the singular configuration of the manipulator. With damping $\lambda = 0$, the damped pseudoinverse is equal to the general pseudoinverse.
It is chosen to make λ adaptive based on how close the manipulator is to a singular configuration. The closeness of the manipulator to the singular configuration is determined by means of singular value decomposition of the Jacobian. In singular value decomposition, a matrix is described as the product of three matrices U, Σ, and V^*.

$$J = U\Sigma V^* \tag{3.31}$$

Σ is a diagonal matrix with singular values. The smallest singular value (Σ_{min}) indicates how close the system is to a singular configuration. The Jacobian is used because its singular values describe how close the manipulator is to a singular configuration.
Using this description of the distance between the manipulators and a singular configuration, an adaptive function for lambda is set up [37]. The chosen adaptive function uses a base damping factor of 0.01 and is described below.

$$\lambda = max([0.1 * \Sigma_{min}, 0.01]) \tag{3.32}$$

The implemented null-space projection has another term that is tuned next to the damping. This is the null-space gain α. The null space gain multiplies the computed actuation. It is desired that this gain be relatively small, as it also multiplies the operational space effects induced by the damping. However, making it too small could result in the null-space policies acting too late. After experimentation, it was found that a α of 0.1 resulted in the best operation.

3.3.5 Null-space policies

The policies implemented for the null-space projections are as follows: Obstacle avoidance, Joint limit avoidance, and null-space damping. Each of these policies results in a desired joint acceleration, which is projected onto the reverence acceleration null-space as described in Equation 3.30. The policies are projected as the sum of all actuations as shown in the following equation:

$$h = \ddot{q}_{\textbf{obstacles}} + \ddot{q}_{\textbf{joint limits}} + \ddot{q}_{\textbf{damping}} \tag{3.33}$$

It was chosen not to implement a task priority projector as described in the theoretical background 2.5.3. This was chosen with the considerations of keeping the projection simple and that the priority in projection is scaled with the exponential nature of the potential fields.

Obstacle avoidance

Obstacle avoidance is the main policy of interest. This policy is tasked with preventing collisions of the manipulator with obstacles in the environment. The obstacle avoidance policy uses the definition of the environment potential field as described in the Obstacle and Potential Field Modelling paragraph. This potential field acts on the manipulator arm at the points of interest as described in the Point of interest paragraph in Manipulator modelling. The gradient vector ($g(x)$) of the potential field is used to describe the force that acts on each point of interest. To be able to use these forces as joint acceleration reverence, they need to be transformed from the operational space to the joint space. This is done by using the respective Jacobian of the link where the point of interest is located. The same Jacobian can be used for each point of interest on the same link, since these links are ridged bodies, and therefore a force can be translated freely over the body. Note however, that this disregards moments induced in the links by offsetting the forces. Therefore, only the Jacobian position is used and not the Jacobian rotation. The point of interest number on the link is n, the manipulator link number is i, and the points of interest are described as $x_{\text{poi,i,n}}$. The transformation of the gradient into angular acceleration is described as follows:

$$\ddot{q}_{\text{obstacle}} = \sum_{i=1} J_i \sum_{n=1} -a \cdot g(x_{\text{poi,i,n}}) \tag{3.34}$$

The gradient vector $g(x_{\text{poi,i,n}})$ is multiplied by $-\gamma$ as the force should act "downhill" of the potential field and a scaling γ is applied to effect the speed by which it should act. γ is tuned to be 1000; this is required to be higher than the joint limit gain as it should act faster. A depiction of these forces induced by an obstacle is shown in Figure 3.8. Since the gradient is used directly as the force that should act on the manipulator, the magnitude and region of effect need to be defined in the potential field. This is done by setting the values of a and d in Equation 3.9. After experimenting, these values were chosen as $a = 0.05$ and $d = 0.75m$.
The joint accelerations generated are added on top of all other null-space accelerations. To guarantee that obstacle avoidance takes priority, a potential field function is used that explodes to infinity when reaching the obstacle wall. Therefore, it takes priority if other null-space actuations force the manipulator close to the obstacle. Note however, that the joint limit avoidance function as described in the following paragraph also explodes up to infinity. meaning that they both may conflict.

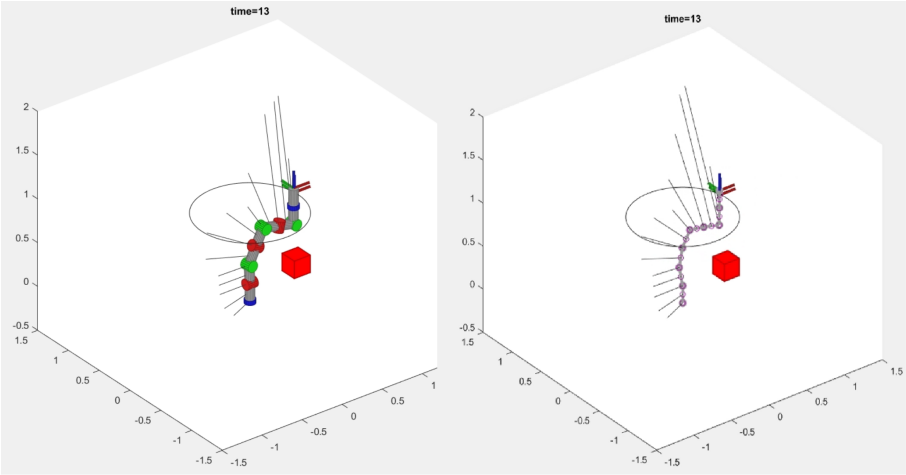

Figure 3.8: Depiction showing a manipulator being pushed away from a obstacle with lines representing the force vectors and a skeleton view showing the points of interest

Joint limit avoidance

Joint limit avoidance is the second important null-space policy used for the manipulator. The joint limit avoidance policy generates reverence accelerations ($\ddot{q}_{\text{joint limit}}$) that move the joints away from the limit. This actuation should only act close to the limit to prevent influence in normal operation, in addition, it should explode to infinity as it approaches the limit to ensure that priority is taken. This can be in conflict with obstacle avoidance as both a joint limit and an obstacle are reached. However, this is accepted, as reaching a joint limit would result in loss of a degree of freedom braking. This would likely result in a loss of tracking performance. A function for the joint limit actuation is constructed that reaches infinity for the negative joint limit (q_{min}) and negative infinity for the positive joint limit (q_{max}) while being close to 0 elsewhere. This function is as follows:

$$\ddot{q}_{\text{joint limit}} = -a \frac{\left(q + \left(\left(\frac{(q_{min}-q_{max})}{2}\right) - q_{min}\right)\right)}{\left(\left(\frac{(q_{min}-q_{max})}{2}\right)^2 - \left(q + \left(\left(\frac{(q_{min}-q_{max})}{2}\right) - q_{min}\right)\right)^2\right)^{\frac{3}{2}}} \tag{3.35}$$

In this function a is the parameter that determines how fast to act on the limit and is tuned to be 100. This gain is chosen lower than the gain of the obstacle avoidance to give it lower priority/tolerate closer limit approach. The function with the limits used for an x / y rotation joint is shown in 3.9. If the joint limit limits are exceeded, which is possible due to the implemented collision model), the actuation is set to 5000 as the function has no solution in that domain.

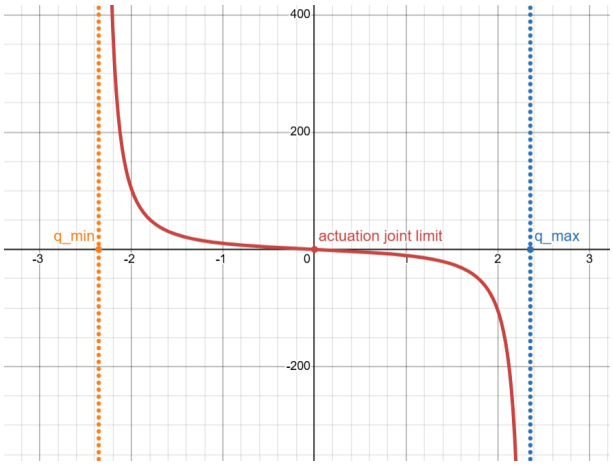

Figure 3.9: Function plot depicting the joint limit actuation function

Null-space damping

Null space damping is an additional null-space actuation term used to remove undesired null space motion. If the null space is not actuated, e.g. no joint limits are reached or no obstacles are close, the null-space of the actuator is free to move. This can result in a build-up of oscillations that can become unstable or build-up significantly so that it influences the operational space. The damping is implemented as a negative gain G_d that penalises joint motion ($\dot{q}$).

$$\ddot{q}_{\text{damping}} = -G_d \cdot \dot{q} \tag{3.36}$$

The gain G_d is tuned to 800 for this implementation.

3.3.6 Inverse kinematics

For the Joint space controller, the operational space trajectory needs to be transformed into a joint space trajectory. This is completed by the inverse kinematics scheme. The chosen inverse kinematics scheme is the optimisation-based approach, as it offers the most accurate performance with the freedom to set boundary conditions.

Minimisation function

The optimisation-based approach optimises for $\dot{q_d}$ so that it minimises the error between a reverence velocity x_{ref} and the transformed joint velocity $J\dot{q}$. This minimisation is executed for each step on the trajectory.

$$\min_{\dot{q_d}}(||J\dot{q_d} - \dot{x}_{ref}||^2 + w \cdot ||\dot{q_d}||^2) \tag{3.37}$$

In addition to the primary minimisation task, a penalty is added to the large joint velocity (Tikhonov regularisation [38]) which reduces the velocity spikes.
The reverence velocity $\dot{x}_ref$ is a combination between the desired velocity of the trajectory $\dot{x}_d$ and an error compensation term. The error compensating term is required since the minimisation assumes that the Jacobian is constant for the time step. This is not the case, as during the time step q varies and the Jacobian is dependent on q. This discrepancy results in a small drift over time if not compensated for. The error is defined as the difference in position for translation and the quaternion error for orientation between the desired previous pose $x_{\text{d prev}}$ and the current pose x.

Another influence on the reverence velocity, is the switching of orientation control. If the orientation control is disabled. both the Jacobian as the reverence velocity only consider translation.

$$\dot{x}_{ref} = \begin{cases} \dot{x}_d + \begin{bmatrix} (x_{\text{d prev}} - x)/ts \\ K_r(\eta_d\epsilon - \eta\epsilon_d + \epsilon_d \times \epsilon) \end{bmatrix}, & \text{if do rotation} = 1 \\ \dot{x}_{\text{d pos}} + (x_{\text{d pos prev}} - x_{pos})/ts, & \text{if do rotation} = 0 \end{cases} \tag{3.38}$$

The ts in the position error compensation is the sample time of the trajectory. This is used to translate the error into a desired velocity to compensate for the error.
Similarly to the velocity-based controller, it was found that the switching of the orientation control caused big spikes in actuation, and for the inverse kinematics even unstable behaviour. Therefore, a similar upper bound was set for the orientation error as described in 3.15. Another similarity to the operational space controllers was the choice to use a scaled Jacobian as shown in 3.13. This was implemented for the same reason as in the operational space control to balance the mismatch in rotation units and translation units.
In this project, the minimisation was done using Matlab's Fmincon function. Using Fmincon, additional constraints can be used to prevent obstacle or joint limit collisions.

optimisation constraints

The minimiser was given both linear constraints for the acceleration and joint limits, as well as non-linear constraints for obstacle collision. The acceleration limit on the joints was set to a fixed value of $40rad/s^2$. The joint limit is defined as an upper and lower bound on the joint position. Both of these linear constraints are implemented as bounds on the joint velocity by means of numeric integration and differentiation.

$$\dot{q}_{ub} = max([40 * ts + \dot{q}, (q_{max} - q)/ts]) \tag{3.39}$$

$$\dot{q}_{lb} = min([-40 * ts - \dot{q}, (q_{min} - q)/ts]) \tag{3.40}$$

Obstacle avoidance is added as a non-linear constraint on the position of the points of interest. Given a function $P_c(q)$ that returns 1 if there is a collision with the environment and 0 if there is no collision, the collision check function can be defined as the function $P_c(q + \dot{q}_d \cdot ts)$. in this function $q + \dot{q}_d \cdot ts$ gives the next estimated q for a given q_d (which is optimised).
However, during the testing, it was found that these conditions were not sufficient to make a successful inverse kinematic scheme. As these boundary conditions only act when reached, the manipulator was often in an optimal solution in a boundary condition or next to an obstacle. This resulted in having no valid solution in the next time step. A common technique used to solve this problem is the null space projection [6].

Inverse kinematics null-space projection

Similarly to operational space control, null-space projection can be used as a method of obstacle avoidance next to the minimiser constraints. This allows the minimiser to act on obstacles and limits before reaching them. The projection of null space is implemented as an additional velocity next to the resulting velocity from the minimiser $\dot{q}_{opt}$. Like the operational space controllers, a damped pseudo-inverse null space projector is used.

$$\dot{q}_d = \dot{q}_{opt} + \alpha(I - J^+J)h \tag{3.41}$$

The damping of the Jacobian pseudoinverse is made adaptive following equations 3.29 and 3.32. This is done to make the behaviour handle singular configurations, while keeping influence on the end effector to a minimum elsewhere.

The big difference between this null space projection and the operational space projection is the domain of projection. As the operational space controllers project onto the joint acceleration, the inverse kinematics implementation projects onto the joint velocity. This requires some adaptations to the null-space policies. The two null-space policies that are implemented are obstacle avoidance and joint limit avoidance. Null-space damping is not implemented as reducing the null-space motion is already handled by the Tikhonov regularisation in the optimisation function. The operation principles of the policies are kept the same as described in the paragraph Null-space policies. The policies are made compatible with the velocity projection by modifying the tuning parameters. For joint limit avoidance, the gain a in the function describing the actuation due to joint limits 3.35 is changed to 1. For the obstacle avoidance policy, the gain was also adapted. The gain a in the obstacle gradient function (equation 3.34) was changed to 10.
The final velocity trajectory q_d including the projection of the null-space is numerically integrated and differentiated to obtain the desired acceleration $\ddot{q}_d$ and position q_d.

3.4 Simulation process

In this section, details are given on how the models are simulated. All simulation and modelling of the test cases is conducted using MATLAB code. There are multiple steps to the process of simulating the cases. The first step is the generation of the manipulator model based on a structure description. This structure description includes the positions of the links, the inertia matrices of the links, and what type of actuator connects this link. The resulting model gives access to the kinematics, dynamic matrices, and configuration of the manipulator.
The second step is to prepare the manipulator for simulation. This includes setting the initial state of the manipulator to zero ($q = 0 \& \dot{q} = 0$). With the exception of linear actuators, which start with an extension of 0.2 metre to prevent starting at the joint limit.
The third step is the generation of the environment. The initial configuration of the manipulator is used in combination with a randomizer seed and a case number to generate a test scenario with trajectory. The starting configuration of the manipulator is required as it determines the starting position of the trajectory. The randomiser seed is used to randomise the case as described in paragraph 3.2.1. This seed is set to 1 for the first simulation of all different cases. For the following iterations, the seed is incremented by 1 to generate different scenarios. The case number used in the generation signals what case to execute. For each iteration, all case numbers are simulated.
The fourth step is to implement the inverse kinematics scheme on the generated trajectory. This inverse kinematics trajectory is also used as a validation test in the scenario. This test is required because it is unknown whether the generated trajectory is realisable by the manipulator. In addition, no easy test is available to test whether the generated trajectory has a valid manipulator joint trajectory with no collisions. By testing if the inverse kinematics is possible, an assurance can be given for at least one valid joint trajectory. If no valid joint trajectory is found, the seed is incremented with 1000, (way above the expected number of tests to prevent duplicates), and a new scenario of the same type is tested. There is one disadvantage using the inverse kinematics as a validation test. By using the inverse kinematics as a validation, a bias is formed towards the inverse kinematics approach.
The fifth step is the simulation of the relevant controllers. All controllers are tested in sequence on the same scenario. An exception is the potential-field guided controller, which is only simulated for the point-to-point cases (agriculture, warehouse, and simple obstacle). The trajectory resulting from case generation is sampled at the intervals of time ts. The entire system is simulated from the current time step t to the next time step $t + ts$. At each time step, the simulation uses the corresponding sample of the trajectory. The simulation is executed using Matlab's ode113 solver. To evaluate the system model (Equation 2.1) using an ode solver, the system dynamics needs to be rewritten in state space form, with the state vector $[q, \dot{q}]^T$. The ode solver

uses this states derivative to compute the next state using numeric variable step integration. The state evolution equation is shown in equation 3.7. The simulation ends when the trajectory is completed. After that, the results are stored for later evaluation. The potential field guided approach is simulated until the goal is reached with signifiant accuracy and the manipulator has held that position for 1 second. In case the controller does not reach the end position, the simulation is terminated after 6 seconds. Which is considered more time than available for the controller.
A flow chart showing the execution of the test is shown in Figure 3.10. For each test case, 20 different scenarios are generated. Each of these scenarios is simulated using each type of controller.
The complete process is then repeated for the three different manipulator models discussed in paragraph 3.1.3

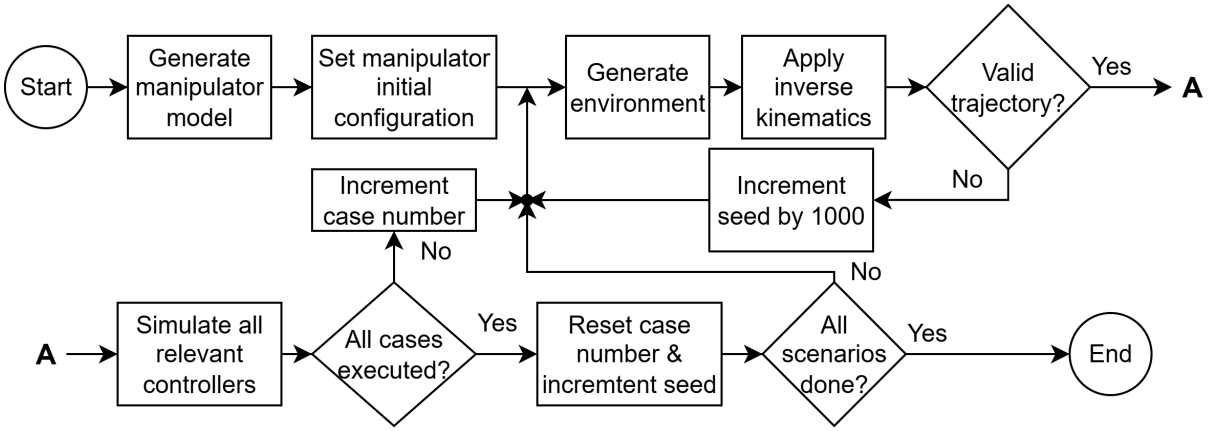

Figure 3.10: A flowchart showing the steps taken during simulation

For rapid testing of the gradient guided control and the local minima estimator, the test cases were evaluated using a gradient descent ode. The state of the system is the position in the operational space, the evolution of which is described by the desired velocity equation 3.18. By solving this ode, a simulated gradient decent for a perfect controller can be tested without simulating the system dynamics. This is used to tune the parameters of the local minima estimator and to rapidly evaluate the performance.

3.5 Evaluation process

In order to get the validation criteria from the simulation data, some processing is required. This section discusses how this processing is conducted. The validation criteria to be evaluated are tracking performance, computational cost, obstacle collisions, and joint collisions. The measures that need to be processed into the validation criteria are in the form of an array describing the state of the manipulator at each time step and a description of all obstacles in the environment.
The tracking error of the manipulator is evaluated by comparing the norm position error of the manipulator end effector with the trajectory at each time step. The position of the end effector can be reconstructed using the manipulator kinematics model and the state at each time step. The orientation error is derived using the norm of the quaternion error described in the theoretical background equation 2.23.
The computational cost is measured differently per section of the test case. The trajectory planning and the inverse kinematics costs are evaluated as the average compute time per time sample. This is the chosen evaluation method as it removes the dependency on the length of the trajectory. The compute time of the controllers is measured with separate tests. This is

done because it is difficult to extract exact information other than the systems state as a function of the change in state from the ode solver, it was chosen to evaluate the compute time of the controllers as an average over a limited number of control cycles. Furthermore, the control computation timing was evaluated as if the controller were implemented sample-based, as would be the case in the real world. The compute time for the estimation of the local minima was also calculated as an average over multiple samples.
Obstacle and joint limit collisions are evaluated by adding two flag states to the model that is simulated. This is possible since it is not required to extract accurate values, but only a true false statement. If a collision occurs during simulation, the state derivative associated with that collision is set to non-zero. The collision information can then be extracted by evaluating if there is a change in the collision states.

4 RESULTS

In the results section of the report, all important findings are discussed. The main interests are as follows. First, the position tracking of the different controller types is evaluated. This is one of the main evaluation parameters, as it gives a measure to evaluate the main goal of the controllers. The second parameter for evaluation is orientation tracking performance. Similarly to the position tracking parameter, it gives a measure of one of the main functionalities of the controllers. However, this measure is only active for smaller sections of the trajectory. Third, the computational costs of the controllers are evaluated. The computational cost is relevant, as it is the main parameter where it is expected that the operational space and gradient-guided approaches have the advantage. The fourth evaluation parameter discussed is the successful run analysis. In the successfully run analysis, key failures are assessed. These key failures are caused by not reaching the end position or by collisions with obstacles. The last parameter for evaluation is overall the performance of the potential-field guided approach. This includes the efficiency of the trajectory and an assessment of the success rate of the local minima estimator.

4.1 Position tracking

For evaluation, the tracking error is defined as the average Euclidean distance between the actual position x and the desired position x_d. The average error is taken for each sample. The position tracking error is evaluated for all successful scenarios, with the exception of gradient-guided scenarios. Gradient guided scenarios are not evaluated for tracking error as they do not have a predetermined trajectory. A successful scenario is defined as a scenario in which no obstacle collision has taken place and the goal position is reached with an accuracy of less than 10 cm. The average position tracking error of the manipulator for each case is shown in Figure 4.1. In this figure, the average tracking error is represented as the average per sample over all runs of a test case.

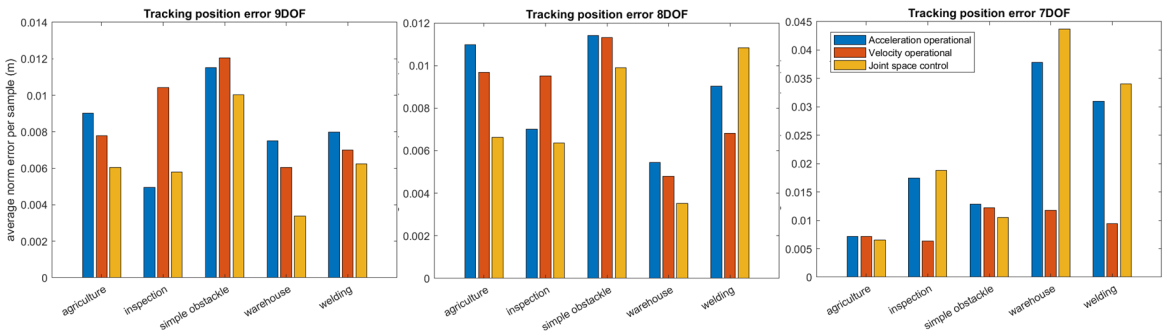

Figure 4.1: The average tracking error per sample for all test cases and manipulator models

As expected, for the 8DOF and 9DOf manipulator models, the operational space has the best tracking performance in the majority of the cases. However, in the 7DOF case, the joint space controller performs the worst and is only the best. In the agriculture and the simple obstacle

cases. Another observation in the 7DOF case is the general increase in the tracking error. This increase in tracking error can be attributed to lower degree of freedom. The lower degree of freedom reduces the ability to move away from obstacles. It is observed that the manipulator therefore operates closer to obstacles. Operation close to obstacles increases the null-space actuation. Since a damped null space projector is used, a higher null space actuation also increases error in the operational space.

The reason why the joint space controller is being outperformed by the operational space controllers in the lower degree-of-freedom cases is not exactly known. One of the theories is that both use different null-space policies. The operational space controllers have an acceleration-based null-space actuation and the joint space controller has a velocity-based actuation in the inverse kinematics scheme. Since null-space actuation is the main source of error, differences in the null-space actuation intensity could cause different errors. This could be tested by evaluating the relative intensities of the null-space actuations and controller actuations, for each controller.

A counter argument could be made for the null-space actuation induced error by looking at the velocity-based control. If the error was induced by the null-space controller, an equal scaling in error would be expected in the acceleration and velocity operational space controllers. This would be the case, as they both use the same projector and policies. However, the tracking performance of the velocity-based controller stays relatively constant for all manipulator models.

Notable is that the velocity-based controller outperforms the acceleration-based controller in almost all cases and models. The acceleration-based controller takes into account the acceleration of the trajectory, which is not done by the velocity-based controller. Another difference between the velocity-based controller and the acceleration-based controller is that the velocity-based controller has an error compensating term in both the joint space and operational space, while the acceleration-based controller only has compensation in the operational space. It is therefore expected that this additional error compensation term in the joint space offers a significant improvement in tracking performance and outperforms the error caused by neglecting acceleration in the trajectory. However, this is only for the current cases considered where the acceleration is kept relatively low (max $\approx 2m/s^2$). In cases with higher acceleration, the error due to acceleration could outweigh the error due to missing compensation in the joint space.

Another observation is the increase in the error of the position of the acceleration-based controller when orientation is enabled. This does not occur for the velocity-based with the same magnitude. This transition is shown in Figure 4.2. This could be one of the contributing factors to the reason why the velocity controller outperforms the acceleration-based controller.

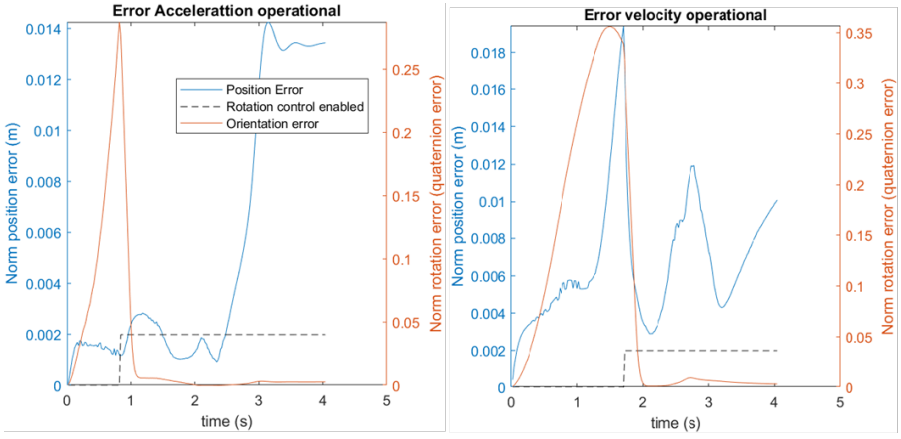

Figure 4.2: Plot showing the destabilisation of position control when orientation control is enabled, (example for warehouse case)

The evaluation of the tracking error shown in Figures 4.2 and 4.1 is based on samples. Since the controller is implemented continuously and the trajectory is zero order held for the continuous time in-between samples, it may be of use to evaluate the influence of the steps in the trajectory and their influence on the tracking error. This error between samples is shown in Figure 4.3 for an agriculture case with a velocity operational space controller. The figure shows an increased error at the start of a sample and a lower error at the end of the sample. This would be expected as the controller receives a new set-point at the start of the sample, which it was not allowed to control for. However, it is important to note that the measurement of the error is done using the end of the time sample using the current set-point. And therefore, the lower error is represented in the sample-based plots. The same process is done for the evaluation of the orientation tracking error.

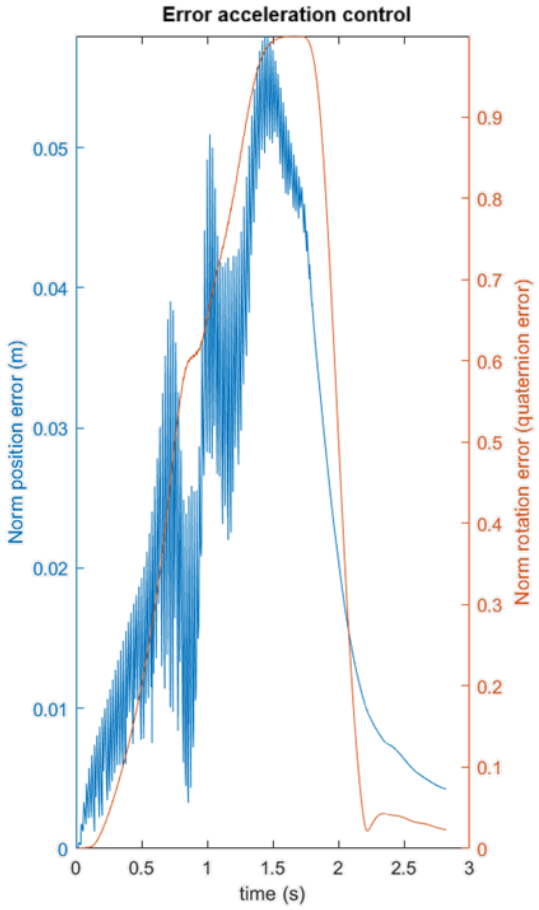

Figure 4.3: The tracking error shown between samples, where the controller still operates

4.2 Orientation tracking error

The orientation tracking error of the manipulator is measured by the norm of the quaternion error per sample. Similarly to the operation space evaluation, the average error per sample was considered per case. This allows us to see in what scenarios the orientation control worked the best. The orientation control error is evaluated only for samples in which the orientation control was enabled. The results are shown in figure 4.4. For reverence, a single-axis 45 degree rotation from the goal orientation would result in a norm quaternion error of 0.7071.

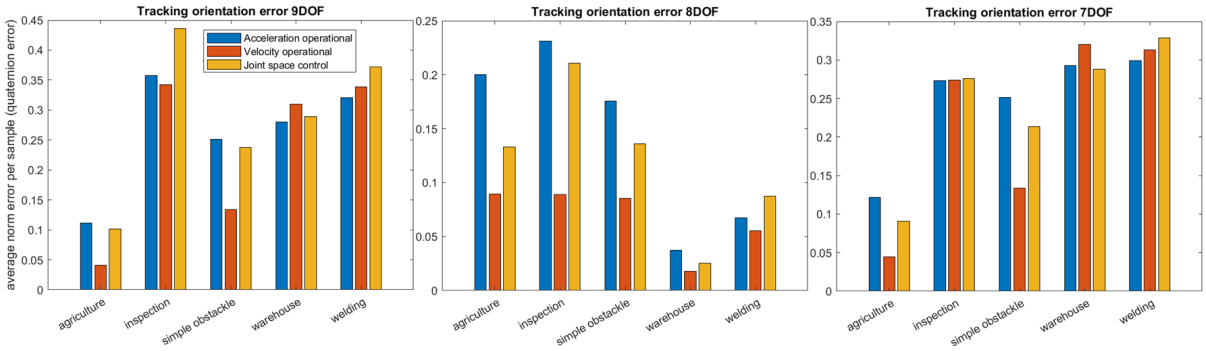

Figure 4.4: Average orientation error of the control types for all scenarios

The first observation is that the joint space controller is not the best performing controller. It would be expected to be the best performing controller due to the inverse kinematics trying to exactly match the given orientation trajectory. One of the theories why the orientation controller is outperformed is that the inverse kinematics compensates the error per step by only a gain in the position error. In contrast, the operational space controllers compensate for position and velocity error. This does not show up in the equal tuning of all the controllers shown in Figure 3.6, as the tuning considers a static desired position. Re-tuning the controllers for a dynamic set point could give a fairer comparison. Alternately, the controllers could be tuned to have bounded actuations, which would be more realistic.
The second observation is that in the best performing controller is the velocity operational space controller. This is unexpected because, as explained in the position tracking error evaluation, the velocity-based controller does not compensate for acceleration. The same argument as used in the position tracking can be used here. That the error compensation in both the operational space and joint space result in a more robust performance.
The one case in which the joint space controller outperforms the operational space controllers most of the time is the weld case. This case is unique in that orientation control is enabled for the entire trajectory. Therefore, the optimiser in the inverse kinematics scheme is capable of closely matching the orientation trajectory, and no large errors need to be compensated for. However, when enabling orientation control from a large orientation error, the optimiser cannot match the trajectory as it needs to first match the trajectory. This large error compensation is theorised to be the part where the velocity control outperforms the joint space controller.

4.3 Computational cost

The computational cost of each of the controllers is evaluated on the basis of the compute time per sample for each of the cases. The compute time is evaluated separately for different sections of the execution. These sections are the Controller computation, Inverse kinematics computations, and path planning computation. The sections are evaluated separately, as they impact some cases more than others. For example, operational space controllers do not require inverse kinematics, and the velocity-gradient approach does not require path planning. An example of all timings is shown in Figure 4.5, displaying the timing for each aspect of control for the 9-DOF model. The timing is given as the computation time per sample in seconds. By evaluating the timing per sample, the distance is removed as an aspect. The timing per sample can then be multiplied by the average number of samples for each test case to get an overall computational cost of an approach per test case.

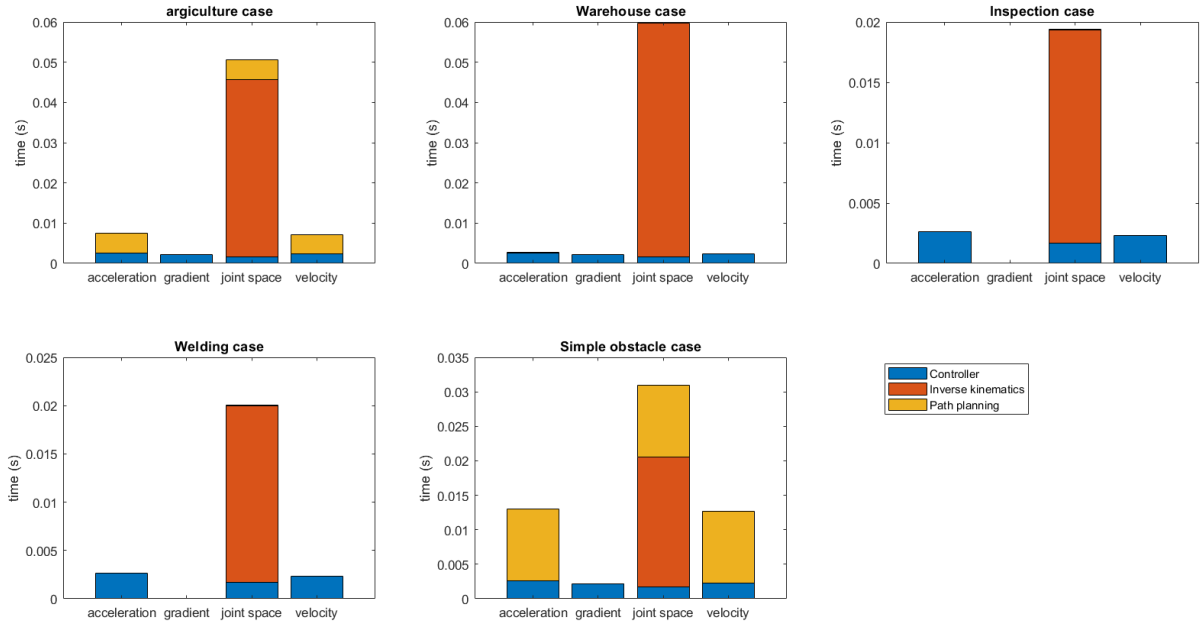

Figure 4.5: The average computational per sample for each test case, measured as the computation time (9DOF model)

The compute time for each controller is really similar. Acceleration-based operation space controller, velocity-based controller, joint space controller, and gradient guided controller have computation times of 0.0026, 0.0022, 0.0017, and 0.0023 seconds, respectively. The difference between the controller compute cost for each model was found to be insignificant. These are small computation costs with respect to the inverse kinematics cost and the path-planning costs for some cases. The timing costs for the inverse kinematics and path planning cases are case-dependent, as can be seen in Figure 4.5.
As can be seen, there is a big difference between the computation time of the cases. This is because the Agriculture and Simple Obstacle cases use a complex RRT path planning algorithm, while the Welding, Inspection, and Warehouse cases use a simple geometric trajectory. One observation is that path planning takes more time in the Simple obstacle case than in the Agriculture case, in contrast the inverse kinematics cost more for the agriculture case than for the simple obstacle case. This is because the inverse kinematics scales linearly with the length of the trajectory, and the RRT path planning scales exponentially with the length of the trajectory. So, whilst the Agriculture case is shorter, it has more obstacles making inverse kinematics expensive. The Simple obstacle case has a longer trajectory, and the inverse kinematics is cheaper due to the reduced number of obstacles. The length of the trajectories is shown in Figure 4.6, as the number of samples.

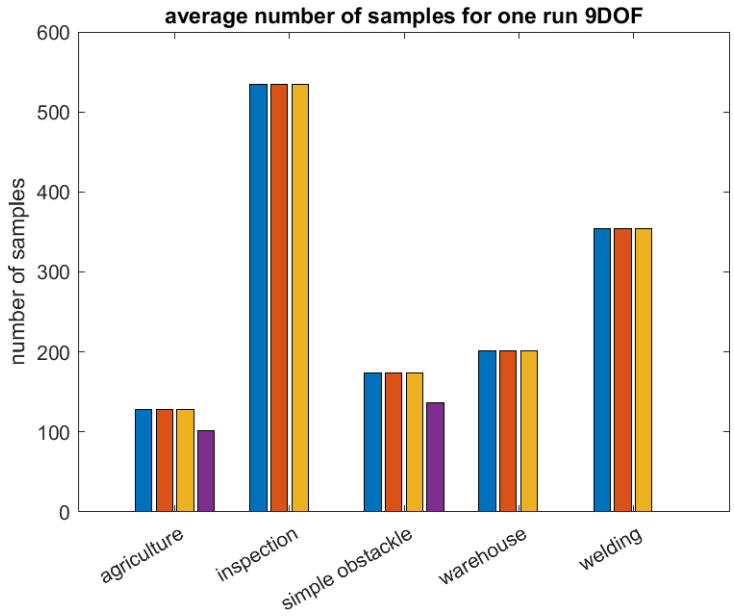

Figure 4.6: Bar graph showing the average number of samples per test case for the 9-DOF model

All controllers except the gradient-guided approach are restricted to the planned trajectory of the path and therefore have the same number of samples. The total computation cost is then computed as the product of the average number of samples and the average compute time per sample. The resulting compute times are shown in Figure 4.7. This figure also shows that there are no significant differences in the computation time for the different models.

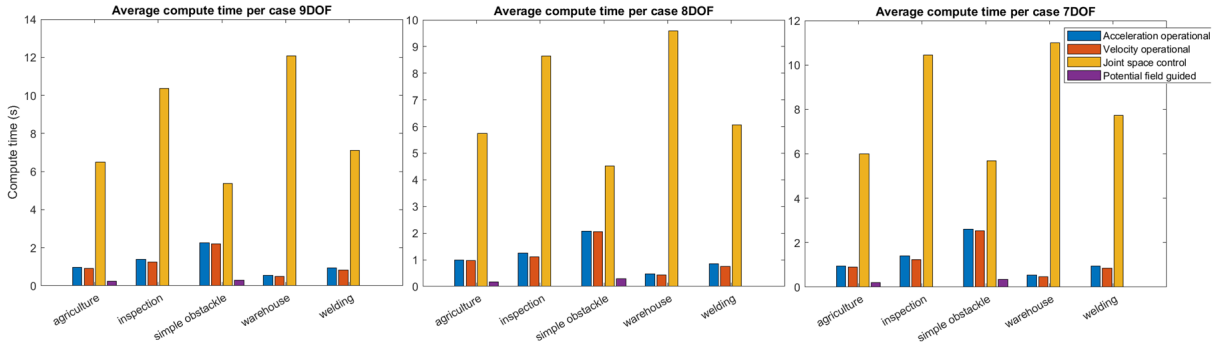

Figure 4.7: The average controller computation time per test case

As can be seen, the joint space controller has the worst performance for the compute time. This is followed by acceleration bases, velocity-based, and velocity gradient. The velocity and acceleration-based operational space controllers are a close match due to the difference being only the small controller compute time difference. The gradient-based approach is the best when considering computation time. However, it lacks significantly in successful runs.

4.4 Successful run analysis

For each test case, controller, and model combination, the number of failed runs is recorded. A run fails if the manipulator collides with an obstacle or if the manipulator does not reach the goal position. These cases are considered failed runs as the primary and secondary tasks are not completed. The failed runs are noted as a percentage of the total number of runs. A

complete overview of the failed runs is shown in Figure 4.8. The main observation is that only the operational space controllers have failed. All failed runs of the velocity- and acceleration-based controller are due to collisions with obstacles. This collision then resulted in some of these cases not reaching the end goal. For many applications, these collisions cannot be tolerated. Showing one of the main reasons why operational space controllers are not often implemented. All the failed runs for the gradient guided approach are due to the end effector not reaching the goal position. For the agriculture case, this is due to an offset in global minimum in potential, and for the warehouse case, this is due to the global minima being obstructed by many obstacles. This is discussed in detail in the Local minima estimation performance paragraph below.

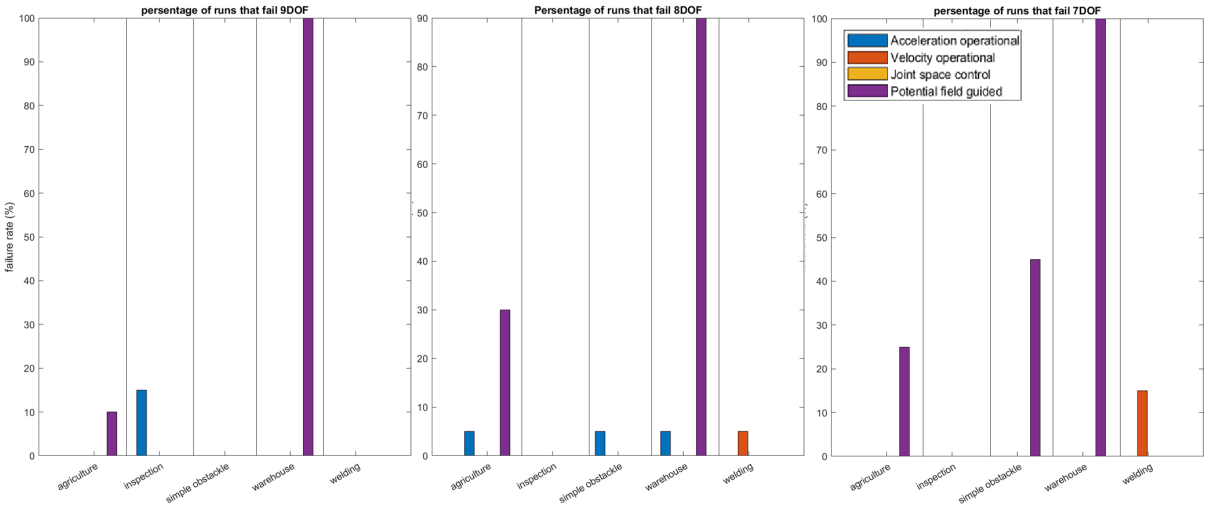

Figure 4.8: Bar graphs showing the Failure rate of each type of controller per case, for all models

4.5 Local minima estimation performance

The gradient decent approach in combination with the novel local minima estimator is accessed by evaluating in how many of the cases the manipulator was capable in reaching the goal position. For the agriculture case, the gradient-based approach only reached the goal position on average for 82% of the time. In the other 18% cases, the end effector did not reach the goal. After evaluation of the potential field, it was found that the obstacle potential field close to the goal position had significant overlap with the goal position. This overlap caused the global minimum to shift. This is shown for an agriculture scenario in Figure 4.9. In some scenarios, this shift was larger than the 10 cm. This distance is used as a measure to detect if a manipulator has reached the goal. As the manipulator stops at this minimum, the scenario fails. A possible solution that could solve this problem is to apply scaling to the different potential fields depending on the distance of the end effector to the goal position. By reducing the obstacle field or increasing the goal attractor, the global minimum can be shifted back to the goal position.

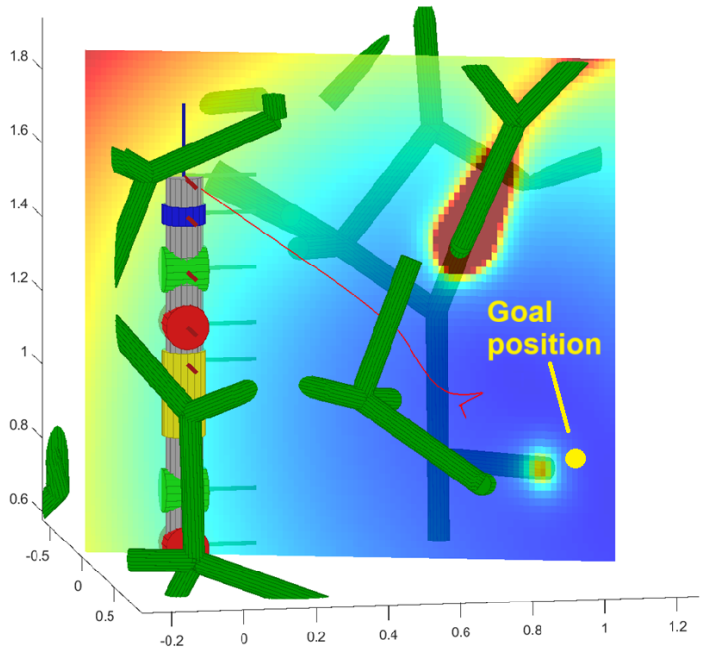

Figure 4.9: An example of a agriculture case with potential field guidance failing due to the obstacle field intersecting with the goal position

For the warehouse case, almost none of the cases were successful in reaching the goal position. Only for the 8-DOF model, 2 of the 20 scenarios were completed without failing. There is no direct reason found why the 8-DOF model is the only one with successful runs, and this could be due to randomly favourable generated cases with fewer obstacles. All other models had 100% failure rate. After evaluating the potential field around the goal position, it becomes clear why. As shown in Figure 4.10, the goal position is completely obscured by the potential fields of the nearby obstacles. Therefore, the local minima estimator places virtual obstacles in front of the shelve. However, there is no lower potential path towards the goal so the virtual obstacle does not help. This is the case for all warehouse cases. This shows that the potential-field guided approach would not be a valid guiding method without significant adaptation in such obstructed environments. Similarly to the agriculture case, the implementation of dynamic scaling of the potential fields could counteract this problem. Furthermore, virtual attractors could be added in the desired directions [32].

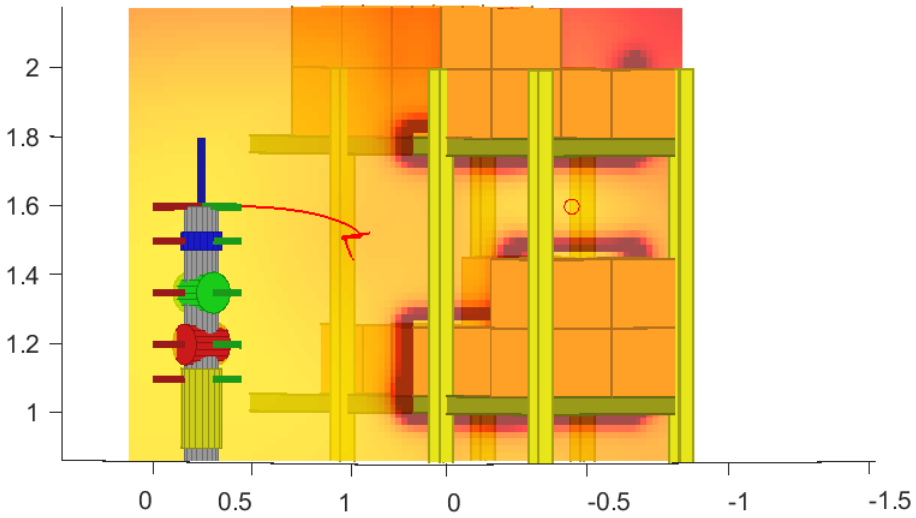

Figure 4.10: Figure showing the potential field close to the goal position (red circle)

The simple obstacle case, in contrast to the agriculture and warehouse case, works as intended. The local minima estimator correctly places obstacles that approach the local minimum. The case is designed so that it is guaranteed that the manipulator moves to a local minimum if it is not counteracted. The early placement of the obstacle causes the manipulator to move around the obstacles towards the goal. An example path taken by the manipulator is shown in Figure 4.11.

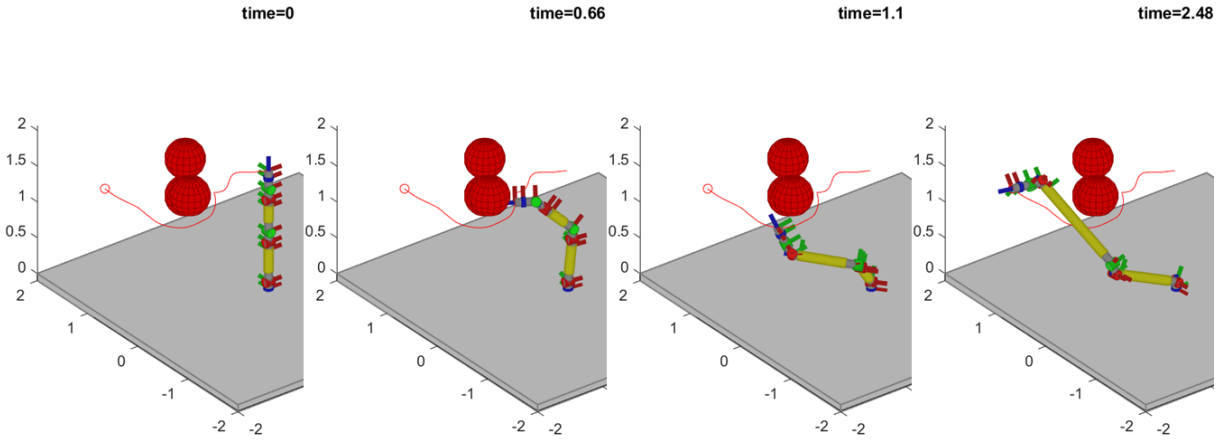

Figure 4.11: Figure showing the gradient guided approach operating in a the simple case scenario

5 DISCUSSION

There are several aspects of the project that need to be taken into consideration. The first is the choice of the environment cases used. The tests are performed in five different cases that are slightly randomised. However, these randomisations are not fundamental to the case description. For example, all warehouse scenarios have two parallel shelves and only a randomised box and a placement of the goal position. The test cases should therefore be considered for the implementation they would represent and not for the general warehouse implementation. For the warehouse case, the idea would be the operation in a structured environment with easily planed geometric trajectories. This brings forth the second point of attention.
The obstacles are all modelled by simple shapes. Therefore, They do not accurately represent obstacles that would be encountered in real-world scenarios. The choice of simple obstacles is made to make the distance calculation efficient. Further simplifications are made in the modelling of the manipulator.
Each link of the manipulator is considered to have the same inertia matrix for simplicity. In addition, the inertia matrix is not changed with the actuation, this would especially be expected from the prismatic joints. This simplification is made to save time on the modelling of the manipulator. This simplification is not considered critical as the operation would not differ significantly as the controllers use the same simplified inertia matrices to cancel out the dynamics.
The trajectory generation for the cases is not guaranteed to succeed in generating a viable trajectory. The method chosen to evaluate the validity of a trajectory is applying the inverse kinematics. If the inverse kinematics is able to find a valid trajectory, the trajectory can be recreated by the manipulator. The problem with this type of evaluation is the introduction of a bias towards the inverse kinematics approach. By only using trajectories validated by inverse kinematics, the joint space controller already has an obstacle-avoiding trajectory, and collision with the obstacles depends only on the performance of the joint controller. In comparison to the operational space controllers, which do not have a guaranteed obstacle collision free path.
The implementation of the controllers that follow these trajectories is also open for discussion. To reduce the time spent on tuning for instability due to time sampling, it was chosen to implement the controllers as time-continuous in the ode solver. By doing this the controllers where easy to theoretically prove to be stable. However, this did remove the time-sampling aspect, making the evaluation more theoretical and less realistic. The continuous implementation of the controller in combination with the sampled trajectory resulted in an additional evaluation complication. The error fluctuated a lot between the start and end of the sample due to the zero-order hold on the trajectory between samples. The choice was made to evaluate the error only at the sample time, as this would give a constant timing to the samples, and the timing is consistent with the start of a trajectory step.
Another problem with the controllers is with how the controllers were made to be comparable. In order to give each controller an equal opportunity of tracking, the controllers were all tuned to have an equal settling time from a step input. However, this is not a real-world limitation that constrains the different controllers. An example of a better method would be the implementation of torque limits on the actuators. This is chosen not to implement as it would require additional null-space actuation to prevent the actuators from stalling, and this would shift the focus from obstacle avoidance. Another problem with the tuning was using a static reverence as step in-

put. This only showed the position error compensation capability of the controllers. A dynamic reverence could be used to tune the velocity error compensation.
One of the main evaluation parameters of the controllers is the computation time. This is measured by evaluating the execution time of the control algorithms. However, there are two problems with this evaluation method. The first is the dependency on how efficient each controller/approach is programmed. Each controller implementation may not have been optimal, and therefore one implementation could have an advantage over the others. However, as the influences are largely dominated by path planning and inverse kinematics with a great margin, some conclusions can safely be drawn. The second problem with execution time is that measurements depend on other tasks running on the computer, which could influence performance. To mitigate this, each test was conducted in order, with no other programmes running.

The last remarks are about the results of the potential-field-guided approach. This approach showed limited success in reaching the goal position. This was due to the implementation of the local minima predictor not being sufficient to solve complex scenarios. This made it look like that the potential field guided approach is not a suitable implementation in any slightly complex case. However, the large computation benefit of not requiring path planning in addition to not needing inverse kinematics, makes this approach well suited to the agriculture case if additions are made.

6 CONCLUSION

The hypothesis that operational space controllers outperform the joint space controller in computation is found to be true. The operational space controllers take more time to execute per sample due to the more complex controller. However, the pre-calculation of the inverse kinematics of the joint space controller is so expensive that the operational space controllers have a overall better performance. The second highest compute cost is due to the the path planning for the cases requiring the RRT algorithm. In these cases (agriculture and simple obstacles), the potential field-guided gradient approach significantly outperforms the other methods.
The hypothesis that the tracking performance of the joint space controller outperforms the operational space controllers was found to be only valid for position tracking for the 9- and 8-DOF manipulator models. For the 7-DOF manipulator model, position tracking was found to be the worst for the joint space controller. This is expected to be caused by the null-space actuation. The null-space actuation for the inverse kinematics may not be equally scaled compared to the operational null-space policies. The lower degree of freedom could force the manipulator closer to obstacles, increasing the null-space actuation and its small influence on the error.
For orientation tracking. The hypothesis that the joint space controller outperforms all other models is false for all manipulator models. It seems that the operational space controllers benefit from the error compensation in the velocity domain, which the inverse kinematics scheme does not apply. This theory is supported by the fact that the joint space control works well for the weld case, where an orientation control is enabled for long sections. This causes inverse kinematics to track small orientation errors in comparison to when the orientation is switched.
The velocity-based operational space controller showed the best orientation control, in addition to being able to handle the lower degree of freedom better. This shows that in the tested cases acceleration compensation in the controller is not more important than the robustness offered by the dual operational space and joint space error compensation. This additionally shows that against predictions, operational space controllers can match the tracking performance of inverse-kinematics driven joint space controllers.
However, operational space controllers come at the cost of reliability. Testing showed that all cases had some failure of one of the control types, except for the joint space controller. The failure rate increased for manipulators with lower degree of freedom, since there is less adaptivity of the manipulator to avoid obstacles. The joint space controller handled this by testing for working trajectories in the inverse kinematics. Therefore, if a valid joint trajectory is found, the joint space controller only needs to closely match the joint trajectory, and a guarantee is given that the manipulator does not collide with obstacles. The advantage of taking the risk of hitting an obstacle is the computational requirements.
The hypothesis that operational space controllers could offer better performance over joint space controllers with respect to computational requirements was found to be true. By not requiring the expensive pre-calculation of the trajectory, the operational space controllers were significantly cheaper to execute on average for all cases.
For cases with a complex path finding algorithm, such as agriculture and the simple obstacle case, it was found that the computational cost could also be reduced by using the potential field guided approach. However, this is combated by the high failure rate of this control method. The best case in which this control method succeeded 100% of the time was the simple ob-

stacle case. In this case, the local minima predictor correctly placed virtual obstacles so that the manipulator would not get stuck and the trajectory would move around the obstacles. For the warehouse case (which would receive minimal benefit from removal of the path finding), it was found that the highly obscured goal position moved the global minimum outside of the shelves. This caused the local minima estimator to place virtual obstacles there, moving the global minima even further from the goal. The result was that only two of the 60 scenarios tested succeeded; this includes all different types of manipulator model. In the agriculture case, which would benefit from the removal of the trajectory planning, there was some failure. These failures were induced by obstacles that were close to the goal position. The potential field of these obstacles interacted with the goal attractor potential field to move the global minima slightly, resulting in the manipulator not reaching the goal position. There are some recommendations on how the guidance could be improved. One of these methods is the implementation of dynamic scaling. Dynamic scaling would adapt the potential field intensities based on how close the manipulator is to the goal position. by reducing the potential field, the manipulator is allowed to move closer to the goal position. Another improvement could be the implementation of virtual goal attractors. By placing virtual goals in front of the manipulator, the trajectory can be forced in a direction. Implementing a compute-efficient solution of these methods could result in the potential field approach being up to $\approx$ 10 times more efficient than joint space controllers, and $\approx$ 3 times more efficient than the other operational space control methods (assuming that these solutions do not add significant computation time and timings are consistent with Figure 4.5).
There are additional recommendations to improve the overall findings in this report. One of the recommendations is to re-tune the controllers and the null-space actuation. Currently, it is unclear whether the null-space policies used in inverse kinematics are equally tuned to the null-space policies used in the operational-space controllers. This could have some effect on the tracking accuracy for the 7-DOF model. Additionally, the controllers could be re-tuned to have a matching performance on velocity tracking next to position. Alternatively, the controllers could be tuned to operate within a maximum given operational performance of the manipulator. This would offer a more realistic comparison between the control approaches. Other recommendations would include solving some of the realism compromises made in the report, as discussed in the discussion. This includes testing on a more accurate manipulator model, including frictions, torque limits, sensor noise, and variable inertia matrices. In addition a more detailed investigation could be done into what takes the highest toll on compute time for the inverse kinematics and path planning. This would give more insight into how efficient the code is implemented.

REFERENCES

[1] Chris Towell, Matthew Howard, and Sethu Vijayakumar. Learning nullspace policies. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 241–248, 2010.

[2] Tsuyoshi Shibata and Toshiyuki Murakami. Motion behavior of null space in redundant robotic manipulators. In Marco Ceccarelli, editor, *Robot Manipulators*, chapter 22. IntechOpen, Rijeka, 2008.

[3] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. In *Proceedings. 1985 IEEE International Conference on Robotics and Automation*, volume 2, pages 500–505, 1985.

[4] Cecilia Scoccia, Giacomo Palmieri, Matteo Claudio Palpacelli, and Massimo Callegari. A collision avoidance strategy for redundant manipulators in dynamically variable environments: On-line perturbations of off-line generated trajectories. *Machines*, 9(2), 2021.

[5] Jaehyung Kim, Wang Jie, Hyunhee Kim, and Min Cheol Lee. Modified configuration control with potential field for inverse kinematic solution of redundant manipulator. *IEEE/ASME Transactions on Mechatronics*, 26(4):1782–1790, 2021.

[6] Jaehuung Kim Sun-Oh Park, Min Cheol Lee. Trajectory planning with collision avoidance for redundant robots using jacobian and artificial potential field-based real-time inverse kinematics. *International Journal of Control, Automation and Systems*, 2020.

[7] Jun Nakanishi, Rick Cory, Michael Mistry, Jan Peters, and Stefan Schaal. Operational space control: A theoretical and empirical comparison. *The International Journal of Robotics Research*, 27(6):737–757, 2008.

[8] Yuki Onishi. An automated fruit harvesting robot by using deep learning, 2019. Accessed: Oct. 2, 2025.

[9] Suplychainbrain. Beyond cost savings: The true roi of warehouse automation, 2025. Accessed: Oct. 2, 2025.

[10] Delwigroenink. Robot welding. Accessed: Oct. 2, 2025.

[11] Aircraft interiors national. Facc evaluates innovative robotic inspection techniques, 2018. Accessed: Oct. 2, 2025.

[12] Rakesh P. Borase, D.K. Maghade, S.Y. Sondkar, and S.N. Pawar. A review of pid control, tuning methods and applications. *International Journal of Dynamics and Control*, 9(2):818 – 827, 2021. Cited by: 739.

[13] David Angeli. Input-to-state stability of pd-controlled robotic systems. *Automatica*, 35(7):1285 – 1290, 1999. Cited by: 106.

[14] Olav Egeland. Task-space tracking with redundant manipulators. *IEEE Journal on Robotics and Automation*, 3(5):471 – 475, 1987. Cited by: 117.

[15] Jun Nakanishi, Rick Cory, Michael Mistry, Jan Peters, and Stefan Schaal. Operational space control: A theoretical and empirical comparison. *International Journal of Robotics Research*, 27(6):737 – 757, 2008. Cited by: 342.

[16] Jan Peters and Stefan Schaal. Learning operational space control. *Proceedings of Robotics: Science and Systems (RSS), Philadelphia, PA*, 01 2006.

[17] O. Khatib and J. Burdick. Motion and force control of robot manipulators. In *Proceedings. 1986 IEEE International Conference on Robotics and Automation*, volume 3, pages 1381–1386, 1986.

[18] B. Xian, M.S. de Queiroz, D. Dawson, and I. Walker. Task-space tracking control of robot manipulators via quaternion feedback. *IEEE Transactions on Robotics and Automation*, 20(1):160–167, 2004.

[19] Alexander Dietrich, Christian Ott, and Alin Albu-Schäffer. An overview of null space projections for redundant, torque-controlled robots. *The International Journal of Robotics Research*, 34(11):1385–1400, 2015.

[20] Xiangyang Sun, Shuai He, Zhenbang Xu, Enyang Zhang, and Yanhui Li. Research on configuration design optimization and trajectory planning of manipulators for precision machining and inspection of large-curvature and large-area curved surfaces. *Micromachines*, 14(4), 2023.

[21] Alexander Dietrich, Christian Ott, and Alin Albu-Schäffer. An overview of null space projections for redundant, torque-controlled robots. *The International Journal of Robotics Research*, 34(11):1385–1400, 2015.

[22] Wenrui Wang, Mingchao Zhu, Xiaoming Wang, Shuai He, Junpei He, and Zhenbang Xu. An improved artificial potential field method of trajectory planning and obstacle avoidance for redundant manipulators. *International Journal of Advanced Robotic Systems*, 15(5):1729881418799562, 2018.

[23] Hang Su, Juan Sandoval, Mohatashem Makhdoomi, Giancarlo Ferrigno, and Elena De Momi. Safety-enhanced human-robot interaction control of redundant robot for teleoperated minimally invasive surgery. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6611–6616, 2018.

[24] N. Oda, T. Murakami, and K. Ohnishi. Null space damping method for robust controlled redundant manipulator. In *Proceedings of IECON'94 - 20th Annual Conference of IEEE Industrial Electronics*, volume 2, pages 755–759 vol.2, 1994.

[25] Mohammed Marey and François Chaumette. New strategies for avoiding robot joint limits: Application to visual servoing using a large projection operator. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 6222–6227, 2010.

[26] Zhao-Cai Du, Guang-Yao Ouyang, Jun Xue, and Yan-Bin Yao. A review on kinematic, workspace, trajectory planning and path planning of hyper-redundant manipulators. page 444 – 449, 2020. Cited by: 21.

[27] Arati S. Deo and Ian D. Walker. Overview of damped least-squares methods for inverse kinematics of robot manipulators. *Journal of Intelligent amp; Robotic Systems*, 14(1):43–68, 1995.

[28] V. V. M. J. Satish Chembuly and Hari Kumar Voruganti. An optimization based inverse kinematics of redundant robots avoiding obstacles and singularities. In *Proceedings of the 2017 3rd International Conference on Advances in Robotics*, AIR '17, New York, NY, USA, 2017. Association for Computing Machinery.

[29] Neil Harrison, Wenxing Liu, Inmo Jang, Joaquin Carrasco, Guido Herrmann, and Nick Sykes. A comparative study for obstacle avoidance inverse kinematics: Null-space based vs. optimisation-based. In Abdelkhalick Mohammad, Xin Dong, and Matteo Russo, editors, *Towards Autonomous Robotic Systems*, pages 147–158, Cham, 2020. Springer International Publishing.

[30] Anoush Sepehri and Amirreza Mirbeygi Moghaddam. A motion planning algorithm for redundant manipulators using rapidly exploring randomized trees and artificial potential fields. *IEEE Access*, 9:26059–26070, 2021.

[31] Iman Soodmand, Maeruan Kebbach, Sven Herrmann, Rainer Bader, and Christoph Woernle. An artificial potential field algorithm for path planning of redundant manipulators based on navigation functions. In Oscar Altuzarra and Andrés Kecskeméthy, editors, *Advances in Robot Kinematics 2022*, pages 470–477, Cham, 2022. Springer International Publishing.

[32] Tao Zhang, Yi Zhu, and Jingyan Song. Real□time motion planning for mobile robots by means of artificial potential field method in unknown environment. *Industrial Robot: the international journal of robotics research and application*, 37(4):384–400, 06 2010.

[33] Bruno Siciliano, Lorenzo Sciavicco, Luigi Villani, and Giuseppe Oriolo. *Robotics: Modelling, Planning and Control*. Springer Publishing Company, Incorporated, 1st edition, 2008.

[34] MATLAB. Ordinary differential equations. Accessed: Oct. 2, 2025.

[35] Xinda Wang, Xiao Luo, Baoling Han, Yuhan Chen, Guanhao Liang, and Kailin Zheng. Collision-free path planning method for robots based on an improved rapidly-exploring random tree algorithm. *Applied Sciences*, 10(4), 2020.

[36] Zhiyuan Xu, Yahui Gan, and Xianzhong Dai. Obstacle avoidance of 7-dof redundant manipulators. In *2019 Chinese Control And Decision Conference (CCDC)*, pages 4184–4189, 2019.

[37] Stefano Chiaverini, Bruno Siciliano, and Olav Egeland. Review of the damped least-squares inverse kinematics with experiments on an industrial robot manipulator. *Control Systems Technology, IEEE Transactions on*, 2:123 – 134, 07 1994.

[38] M.E. Hochstenbach and L. Reichel. An iterative method for tikhonov regularization with a general linear regularization operator. *Journal of Integral Equations and Applications*, 22(3):465–482, 2010.

[39] Kevin M. Lynch and Frank C. Park. *Modern Robotics: Mechanics, Planning, and Control*. Cambridge University Press, USA, 1st edition, 2017.

A TWIST MANIPULATOR MODELLING

The dynamics of the manipulator is modelled using the Euler-Lagrange method. However, the Euler-Lagrange equations require a formulation of the kinetic and potential energy as a function of the generalised coordinates (shown in equation 3.2). These kinetic energy and potential energy functions are derived using the twist method as described in this appendix.

Kinetic energy

The kinetic energy of the system can be analysed for each element i of the number of elements n, using the twist motion ($\xi_i^{0,0}$) (with respect to the inertia frame expressed in the inertia frame) and the inertia matrix (equation A.1). Here, a twist is a six-dimensional vector that describes the complete motion of a body in a 3D coordinate frame with respect to a other coordinate frame. The robot arm modelling by the twist method is performed as described in Modern Robotics [39]. The notation of twist used in this report follows the following convention: $\xi_a^{b,c}$ describes the twist of the coordinate frame a with respect to the coordinate frame c, expressed in the coordinate frame b. The inertial frame is denoted as 0 and forms the basis of the first link.

$$T = \sum_{i=1}^{n} T_i = \frac{1}{2} \sum_{i=1}^{n} (\xi_i^{0,0})^T I_i (\xi_i^{0,0}) \tag{A.1}$$

The above-mentioned twist motion of each element can be computed using the geometric Jacobean ($J_i^*(q)$) of each element and the joint velocities ($\dot{q}$).

$$\xi_i^{0,0} = J_i^*(q)\dot{q} \tag{A.2}$$

Here, the geometric Jacobean exists out of the unit twists of each joint with respect to the previous joint expressed in the inertia frame.

$$J_i^*(q) = [\hat{\xi}_1^{0,0}, \hat{\xi}_2^{0,1}, ..., \hat{\xi}_i^{0,i-1}] \tag{A.3}$$

Unit twists can be computed by an algorithm using the initial pose of the system $H(0)$, the generalised coordinates q, and the unit twist describing the axis of motion in each body with respect to the previous body $\hat{\xi}_i^{i-1,i-1}$ expressed in the frame of the previous body.

The algorithm iterates through each of the ridged moving bodies as follows: The first step is to calculate the homogeneous transformation matrix of the first body using the unit twist, the angles, and the initial system pose.

$$H_i^{i-1}(q_i) = e^{\tilde{\hat{\xi}}_i^{i-1,i-1} \cdot q_i} \cdot H_i^{i-1}(0) \tag{A.4}$$

Note that this transformation is from the previous body to the current body, and not the inertial frame, if the current body is not the first body. If the current body is not the first body, the previously calculated transformations can be used to transform the homogeneous transformation matrix from the current body to the inertial frame.

$$H_i^0(q) = H_1^0(q_1) \cdot H_2^1(q_2) \cdot ... \cdot H_i^{i-1}(q_i) \tag{A.5}$$

These homogeneous transformations describe the kinematics of each of the manipulator joints. In addition, these transformations in combination with the unit twist describing the possible motion between the current body and the previous body expressed in the previous body frame, can be used to calculate the twist describing the motion of the current body in the inertial frame. This can be done as follows where Ad_H is the adjoint of the transformation H.

$$\hat{\xi}_i^{0,i-1} = Ad_{H_{i-1}^0(q)}\hat{\xi}_i^{i-1,i-1} \tag{A.6}$$

These twists can then be used in equation 3.3 to calculate the kinetic energy.

Potential energy

The potential energy in the system V is divined by mass and gravity. The potential energy of each segment can therefore be calculated by multiplying the position vector of the centre of mass (P_{mi})transformed by the robot configuration H_i^0, the gravity vector transposedg^T, and the mass of the segment m_i.

$$V = \sum_{i=1}^{n} V_i = \sum_{i=1}^{n} (H_i^0 P_{mi}) g^T m_i \tag{A.7}$$

Where:

$$P = \begin{bmatrix} P_x \\ P_y \\ P_z \end{bmatrix} \text{ \& } g = \begin{bmatrix} g_x \\ g_y \\ g_z \end{bmatrix} \tag{A.8}$$

The resulting differential equations 3.2 can be rewritten in a form with a mass, Coriolis, and potential energy matrix.

$$\tau = M(q)\ddot{q} + C(q,\dot{q}) + g(q) \tag{A.9}$$