FH University of Applied Sciences TECHNIKUM WIEN

MASTER THESIS

Thesis submitted in fulfillment of the requirements for the degree of Master of Science in Engineering at the University of Applied Sciences Technikum Wien - Degree Program Robotics Engineering

Alogrithmic Payload Estimation

By: Moritz Dönges, BSc

Student Number: 2310331024

Supervisor: Michael Schebeck, MSc

Vienna, July 27, 2025

Declaration

Vienna, July 27, 2025 Signature

Kurzfassung

Im Kontext der digitalen Fabrik an der UAS Technikum Wien, wo Menschen und Roboter sich die Aufgaben und den Arbeitsbereich teilen, ist die sichere und effiziente Handhabung von Nutzlasten von entscheidender Bedeutung. In der digitalen Fabrik der UAS werden Nutzlasten derzeit noch ohne Kenntnis ihrer internen Parameter gehandhabt, was zu potenziellen Manipulationsfehlern führen kann, die Menschen Schaden zufügen. Diese Studie beschreibt die Entwicklung einer fortschrittlichen Methode zur Kraft-/Drehmomentabschätzung, um die Fähigkeit eines UR5-Roboters zu verbessern, verschiedene Nutzlastbedingungen zu erkennen und zu handhaben. Diese Fähigkeit gewährleistet die Wahrnehmung des auf einer mobilen Industrieroboterplattform montierten UR5-Roboters, um den sicheren und effizienten Transfer von Nutzlasten zwischen verschiedenen Arbeitsbereichen innerhalb der Fabrik zu erleichtern. Die modernsten Methoden zur Kraft-/Drehmomentabschätzung für Industrieroboter nutzen neuronale Netze und Gauß-Prozesse als führende Methoden für genaue Nutzlastabschätzungen. Es wurde ein Gauß-Prozess-Modell entwickelt, um die Kräfte und Drehmomente abzuschätzen, die vom Roboter bei der Ausführung von Trajektorien erzeugt werden. In einem zukünftigen Projekt kann das Bewusstsein für Nutzlasten auf dem UR5-Roboter hinzugefügt werden. Auf diese Weise zielt die Studie darauf ab, die Intelligenz von Robotersystemen in industriellen Umgebungen zu verbessern und den Weg für eine höhere Produktivität und Sicherheit in digitalen Fertigungsumgebungen zu ebnen. Dieses Projekt führte auch zu einer Simulation, die eine Grundlage für die Aufzeichnung der Sensordaten aus dem UR5-Interieur.

Schlagworte: Gaussian Process, Force Estimation, Newton/Euler, UR5 Robot, Rigid Body

Abstract

In the context of the digital factory, at UAS Technikum Vienna, where humans and robots share the tasks and the workspace, the safe and efficient handling of payloads is essential. At the UAS digital factory payload is still handled without recognising anything about the payloads internal parameters, leading to potential manipulation failures causing human harm. This study describes the development of an advanced force/torque estimation method to improve a UR5 robots ability to recognize and handle different payload conditions. This capability ensures the perception of the UR5 robot mounted on a mobile industrial robot platform to facilitate the safe and efficient transfer of payloads between different workspaces within the factory. The state of the art methods of force/torque estimation for industrial robots serve neuronal networks and gaussian processes as the leading methods for accurate payload estimations. A gaussian process model has been developed to estimate the forces and torques generated by the robot when executing trajectories. In a future project face, an awareness of payloads can be added on the UR5 robot. In this way, the study aims to improve the intelligence of robotic systems in industrial environments and pave the way for higher productivity and safety in digital manufacturing environments. This project face also yeelted in a simulation that provides a basis to record the sensor data from the UR5's internal sensors and a force/torque sensor and a pipeline to train and evaluate gaussian process models.

Keywords: Gaussian Process, Force Estimation, Newton/Euler, UR5 Robot, Rigid Body

Acknowledgements

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Contents

1 Introduction

Robots play a central role in modern industry and manufacturing, but with increasing complexity of tasks and the need for closer human-robot collaboration, new challenges arise. Safe manipulation of payloads and safe interaction with humans are the two main challenges in human-robot collaboration. Both require the knowledge of the inertial parameters of the payload. The ability to recognize different payload conditions advantages a safer human-robot collaboration. The primary objective of this analysis is to estimate the mass m and inertia tensor J of the manipulated payload, ϕ_{payload} (**??**). The manipulated payloads considered here are geometrically simple shapes—namely, a cube and a cylinder—where the center of mass is located at the geometric center of each rigid body. The contribution of this study is to advance human-robot collaboration safety by enabling robots to estimate the mass and inertia tensor of payloads in real-time. Awareness of these inertial parameters is crucial for mitigating risks and ensuring safe manipulation in close proximity to humans, a priority recognized by recent advancements in human-robot interaction research [**bai2024sensorless**, **Popov2019**, **nadeau2022fast**, **kurdas2022online**, **su2021deep**, **haninger2022model**, **HANINGER2023104431**].

At first there are two notations that are important in this case.

$$\phi^T = \begin{bmatrix} m & mc_x & mc_y & mc_z & & \\ J_{xx} & J_{xy} & J_{xz} & J_{yy} & J_{yz} & J_{zz} \end{bmatrix} \in \mathbb{R}^{10} \tag{1}$$

The vector ϕ describes the internal parameter properties of a rigid body.

$$\begin{bmatrix} \mathbf{f} \\ \boldsymbol{\tau} \end{bmatrix} = m \begin{bmatrix} \mathbf{I}_{3\times3} & -[\mathbf{c}]^\times \\ [\mathbf{c}]^\times & \mathbf{J}_s \end{bmatrix} \begin{bmatrix} \mathbf{a} \\ \boldsymbol{\alpha} \end{bmatrix} + \begin{bmatrix} m[\boldsymbol{\omega}]^\times[\boldsymbol{\omega}]^\times \mathbf{c} \\ [\boldsymbol{\omega}]^\times \mathbf{J}_s \boldsymbol{\omega} \end{bmatrix} \tag{2}$$

The Newton-Euler equations describe how the forces and torques acting on a rigid body are related to its acceleration ($\mathbf{a}$), angular acceleration ($\boldsymbol{\alpha}$) and angular velocity ($\boldsymbol{\omega}$), taking into account its mass properties and moments of inertia.

At this point a effective rigid body is introduced. It is described as $\phi_{\text{effective}}$ (**??**) and is differentiated in ϕ_{payload} and ϕ_{gripper}.

$$\phi_{\text{effective}} = \phi_{\text{payload}} + \phi_{\text{gripper}} \tag{3}$$

Since a force/torque sensor is mounted between the robot flange and the gripper, the Newton/Euler equations (**??**) can be used to determine the internal parameters of $\phi_{\text{effective}}$ [**kurdas2022online**, **nadeau2022fast**]. We can measure the force/torque (f/t) data at the measurement frame (MF) of the f/t sensor and use more sensor data from motor sensors to get $\mathbf{a}$, $\boldsymbol{\alpha}$, and $\boldsymbol{\omega}$. By the Newton/Euler equations the mass m and the inertia tensor J of the payload can be calculated. To get information about the center of mass $\boldsymbol{c}$ and due to having more unknowns than equations in a single measurement in this case, a least squares approach can be used with multiple data sets to estimate $\boldsymbol{m}$, $\boldsymbol{c}$, and $\boldsymbol{J}$, minimizing the overall error and providing an optimal solution [**kurdas2022online**, **nadeau2022fast**].

Handling ϕ_{payload} leads to a f/t-measurement reflecting $\phi_{\text{effective}}$ now. So it clears out ϕ_{gripper} needs to be isolated to get to the $\boldsymbol{m}$ and $\boldsymbol{J}$ of the ϕ_{payload}.

To isolate the force/torque measurements specific to ϕ_{gripper} while the robot carries both ϕ_{gripper} and ϕ_{payload}, estimation is required. Additionally, as the motor sensors register $\phi_{\text{effective}}$ — the combined effect of both gripper and payload — this estimation is necessary to separate these contributions.

Acceleration, angular acceleration and angular velocity show up as independent features in our motion in relation to our ϕ_{payload}, as long as the manipulation of $\phi_{\text{effective}}$ does not exceed the maximum motor loads [**urrea2018parameter**], (Sec. **??**).

In contrast the effort shows up as dependent feature in relation to ϕ_{payload}. The motors efforts increase proportional with increasing ϕ_{payload}, while ϕ_{gripper} always remains the same (**??**), (Sec. **??**).

To isolate the f/t measurements at the MF resulting from ϕ_{gripper} from the actual ft measurement at the MF, the sensor data used must be reflected. The transmission function from the motor torques (that are based on the effort sensor data $\boldsymbol{I}$ measured (**??**), (**??**)) to the resulting actual f/t measurement at the MF (**??**) is non-linear due to the complex interactions within the system (**??**). This non-linearity arises from factors such as the dynamics of joint coupling, friction, and gear reductions, which do not scale linearly with torque. Additionally, the influence of the center of mass and the changing inertia properties of the robot while it moves contribute to this non-linearity (**??**), (**??**), [**liu2019end**, **blumberg2023estimation**].

Taking the non linearity and the fact we need to get the f/t-measurement resulting of ϕ_{gripper}, while moving $\phi_{\text{effective}}$, results in two estimation models.

The effort sensor data shows up as a key feature here because it directly connects f/t, $\boldsymbol{a}$, $\boldsymbol{\alpha}$, and $\phi_{\text{effective}}$ (see Eqs. (**??**) and (**??**), and Sec. **??**).

Now two GP Regression (Sec. **??**) models are trained based on motion data while manipulation with ϕ_{gripper}. The GP$_{\text{effort}}$ model's predicted targets are the effort values of each motor, based on the motion with ϕ_{gripper} [**kurdas2022online**, **urrea2018parameter**]. The GP$_{\text{wrench}}$ model's targets are the ft values at the measurement frame based on the motion with ϕ_{gripper} [**kurdas2022online**, **urrea2018parameter**].

This approach for estimating external forces resulting from a manipulated payload is implemented with a simulated Universal UR5 Robot [**ur5_robot**] in a Robot Operation System (ROS) [**ros_official**] Gazebo [**gazebo_ros**] physics simulation.

2 State of the Art

A focus in the current human-robot collaboration research is external force estimation for collaborative industrial robots research [**bai2024sensorless**, **Popov2019**, **nadeau2022fast**, **kurdas2022online**, **su2021deep**]. Due to the non-linearity in robotic systems, arising from factors such as the dynamics of joint coupling, friction, and gear reductions, various estimation methods have been established. A distinction is made between Classical Estimation Methods and Machine Learning and Neuronal Network Estimation Methods. Classical estimation methods are techniques based on mathematical models, optimization, and statistical analysis used to estimate unknown parameters or states of a system. These methods typically involve deterministic or probabilistic approaches that are derived from physical models, sensor data, and known system behaviors. They do not rely on data-driven learning algorithms but instead use analytical solutions and optimization frameworks to minimize errors or maximize the accuracy of parameter estimation. Classical estimation methods include Optimization-Based Methods (e.g., Least Squares (LS), Weighted Least Squares (WLS), Iterative Reweighted Least Squares (IRLS) [**dong2023dynamic**, **nadeau2022fast**, **golluccio2021robot**, **Xu2022accurate**]), Model-Based and Analytical Techniques (e.g. Momentum Observer [**kurdas2022online**]), and Statistical Techniques (e.g., Kalman Filter, Principal Component Analysis (PCA) [**liu2021sensorless**, **motor_current_estimation**, **berger2021feature**]).

Machine learning and neural network estimation methods use data-driven algorithms to model and predict unknown system parameters or states [**ren2020learning**, **su2021deep**]. These methods learn from historical data and can adapt to complex, non-linear relationships that may be difficult to model explicitly with classical approaches. The use of neural networks allows for pattern recognition and predictions based on input features, without needing explicit programming for each situation [**zeng2019tossingbot**, **Kruzic2021**]. Machine learning and neural network estimation methods include Neural Network Architectures (e.g., Feedforward Neural Networks [**su2021deep**], Recurrent Neural Networks (RNNs) [**berger2016estimating**], Long Short-Term Memory (LSTM) [**berger2021feature**, **Kruzic2021**], Convolutional Neural Networks (CNNs) [**Kruzic2021**, **zeng2019tossingbot**]), Advanced Learning Models (e.g., Generative Adversarial Networks (GANs) [**ren2020learning**], Conditional GANs (CGANs) [**ren2020learning**], Least Squares GANs (LSGANs) [**ren2020learning**]), Probabilistic and Bayesian Methods (e.g., Gaussian Process Regression (GPR [**rasmussen2006gaussian**]) [**haninger2022model**, **beckers2017stable**, **nguyen2008learning**, **HANINGER2023104431**], Hybrid GPR with Joint Stiffness Models [**blumberg2023estimation**], and Experience-Based and Feature Learning (e.g. Experience-Based Torque Estimation [**berger2016estimating**, **berger2015learning**]).

These estimation methods are employed to gather information about various parameters in robotic control and motion. Estimating the dynamic model parameters of robotic systems that are not interacting with their environment focuses on optimizing the robot's trajectory control [**lee2020adaptive**] and torque control [**nguyen2008learning**]. Additionally, understanding the inverse kinematics [**ren2020learning**] and inverse dynamics [**leon2022parameter**, **golluccio2021robot**] of the robotic system (**??**) serves as a foundation for external contact force estimation. Here classical estimation methods are present[**dong2023dynamic**, **golluccio2021robot**, **deng2021dynamic**], as well as machine learning [**nguyen2008learning**] and neuronal network estimation methods [**leon2022parameter**, **urrea2018parameter**].

For external contact force estimation, two key scenarios are considered. The first involves estimating contact forces at the robot's links [**Popov2019**, **su2021deep**], either with [**Popov2019**] or without [**su2021deep**] determining the location of the force on the link's surface, accounting for multiple contact points with the environment. The second scenario focuses on external contact force estimation at the TCP (Tool Center Point), where only the TCP interacts with the environment. TCP external contact force estimations are crucial for trajectory and control optimization [**HANINGER2023104431**, **haninger2022model**], torque control [**motor_current_estimation**], and detecting and minimizing positioning errors [**Tan2023**, **lu2023external**]. This involves estimating $\phi_{\text{effective}}$. Classical TCP contact force estimation [**motor_current_estimation**], machine learning methods [**HANINGER2023104431**, **haninger2022model**, **beckers2017stable**, **berger2015learning**, **blumberg2023estimation**] and neuronal network estimation models [**Tan2023**, **lu2023external**, **Kruzic2021**, **shan2024fine**, **berger2016estimating**, **liu2021sensorless**, **su2021deep**] reaching good results in control optimization and position error detection.

Furthermore, TCP external force estimation is used to gain insights into the internal parameters of the payload. In such cases, the contact force at the TCP is differentiated into ϕ_{gripper} and ϕ_{payload} (**??**). Both classical estimation methods [**Hu2020**, **Xu2022accurate**, **kurdas2022online**, **nadeau2022fast**] and neuronal network estimation methods [**berger2021feature**] are utilized.

Since the dynamic force equation (**??**) defines the internal parameters of the robots rigid body ϕ_{robot} while motion and shows that τ_{motor}, especially $\mathbf{I}$ is the key feature in this estimation approach (**??**) (**??**) (**??**), the Newton/Euler equations (**??**) give the conditions to calculate the internal parameters of a rigid body $\phi_{\text{effective}}$ (**??**) based on the trajectory parameters $\mathbf{a}$, $\boldsymbol{\alpha}$, and $\boldsymbol{\omega}$. The literature has shown that a parameter identification using the Newton/Euler equations leads to valid information of the the rigid body's ϕ_{robot} [**lee2020adaptive**, **nguyen2008learning**, **ren2020learning**, **leon2022parameter**, **dong2023dynamic**, **golluccio2021robot**, **deng2021dynamic**], $\phi_{\text{effective}}$ [**haninger2022model**, **HANINGER2023104431**, **beckers2017stable**, **berger2015learning**, **blumberg2023estimation**, **motor_current_estimation**, **an2023**, **lu2023external**, **Kruzic2021**, **shan2024fine**, **berger2016estimating**, **liu2021sensorless**, **su2021deep**] and ϕ_{payload} [**Hu2020**, **Xu2022accurate**, **kurdas2022online**, **nadeau2022fast**] while motion. Also GP Regression models show their ability to estimate various parameters in this context [**haninger2022model**, **beckers2017stable**, **nguyen2008learning**, **HANINGER2023104431**, **blumberg2023estimation**, **berger2015learning**], which is why it is chosen to overcome the non-linearity of in the robot dynamics with respect to the motor sensor data and the resulting f/t-measurements at the MF and to isolate ϕ_{payload} from $\phi_{\text{effective}}$.

2.1 Background

Linear Acceleration ($\mathbf{a}$), Angular Velocity ($\boldsymbol{\omega}$), and Angular Acceleration ($\boldsymbol{\alpha}$) can be considered independent features with respect to $\phi_{\text{effective}}$ as long as the robot can perform the motion without exceeding its physical limits. These kinematic parameters are set by the robot's trajectory planning and remain the same whether a payload is carried or not, assuming the payload does not exceed the robot's capabilities. The robot can execute the same trajectory with the same velocity and acceleration parameters regardless of whether it is carrying a payload [**urrea2018parameter**].

The motor effort is not independent with respect to $\phi_{\text{effective}}$ because it is affected by the added inertia and gravitational load when a payload is attached.

The General Joint Torque Equation (**??**) is central to Inverse Dynamics of robots because it determines the joint torques necessary for a given trajectory. In this case, let the robot's rigid body be ϕ_{robot}. Then $\phi_{\text{robot}} + \phi_{\text{gripper}}$ are reflected as τ_{robot}, because both remain the same for every motion [**Popov2019**].

$$\boldsymbol{\tau}_{\text{robot}} = \mathbf{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \mathbf{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \mathbf{G}(\boldsymbol{q}) \tag{4}$$

External forces like human interaction or handling payloads acting on the robot and influence the joint torques necessary for a given trajectory. So τ_{ext} is the vector of external torques at the joints due to external forces acting on the robot. The transpose of the robots Jacobian matrix $\mathbf{J}^T$ is used to project these external forces back to the joint space, indicating the torques required at each joint to counteract or respond to those forces. $\vec{F}_{\text{ext}}(\phi)$ is the vector of external forces applied to the robot's end-effector by a payload. [**Popov2019**, **lu2023external**].

$$\boldsymbol{\tau}_{\text{ext}} = \mathbf{J}^T \vec{F}_{\text{ext}}(\phi) \tag{5}$$

This leads to the motors required torques to move the robots rigid body and a payload for a given trajectory [**liu2021sensorless**].

$$\boldsymbol{\tau}_{\text{motor}} = \boldsymbol{\tau}_{\text{robot}} + \boldsymbol{\tau}_{\text{ext}}(\phi) \tag{6}$$

Since the UR5 robot in this case uses brush less DC motors, the torque-current relationship for electric motors expresses the linear relationship between the torque τ_{motor} produced by a motor and the current $\mathbf{I}$ supplied to it [**deng2021dynamic**, **ctms_motor_speed**].

$$\boldsymbol{\tau}_{\text{motor}} = k_t \cdot \boldsymbol{I} \tag{7}$$

Now converting Equation (**??**) to the motor efforts $\mathbf{I}$ and inserting Equation (**??**) we see that the joint motor efforts are proportional related to the payload we add at the robots MF, while τ_{robot} remains the same.

$$\boldsymbol{I} = \frac{\boldsymbol{\tau}_{\text{robot}} + \boldsymbol{\tau}_{\text{ext}}}{k_t} \tag{8}$$

The resulting motor torques that are applied to move the robot and the rigid body of the payload result in the f/t measurements on the MF. Where the transmission function from the motor torques to the resulting ft measurements at the MF is non-linear due to the complex interactions within the robot's system. This non-linearity arises from the dynamics of joint coupling, friction, and gear reductions that do not scale linearly with torque. Additionally, the influence of the center of mass and the changing inertia properties as the robot moves contribute to this non-linearity (**??**) and the external torque projection (**??**) highlight the non-linear dependencies on the joint angles $\mathbf{q}$, velocities $\dot{\mathbf{q}}$, and inertial properties ϕ. This relationship can be described with:

$$\vec{F}_{\text{measured}} = f(\tau_{\text{motor}}, \mathbf{q}, \dot{\mathbf{q}}, \phi) \tag{9}$$

where $f(\cdot)$ is a non-linear function that depends on the joint configuration, velocities, and inertial parameters ϕ. These factors make the mapping from motor sensor data (effort, position, velocity) to resulting ft measurements at the MF complex and non-linear [**liu2019end, blumberg2023estimation**].

The Motor effort (τ_{motor} (**??**)) is central to understanding the relationship between motor torques, rigid body motion, and the resulting force/torque ($\vec{f}$ and $\vec{\tau}$) as described by the Newton-Euler equations. τ_{motor} represents the torque produced by the motors, which is applied to move the robot's joints and the connected rigid body. The Newton-Euler equations describe how forces and torques acting on a rigid body are related to its mass properties and motion (**??**). The torque exerted by the motor (τ_{motor}) drives the joints and is responsible for creating motion in the connected rigid body. This torque results in joint accelerations ($\vec{\alpha}$), which translate to angular accelerations of the rigid body. The motion described by linear acceleration ($\vec{a}$) and angular acceleration ($\vec{\alpha}$) depends on the distribution of mass (m) and the inertia tensor ($\mathbf{J}_s$) of the body. When the motors apply more torque (τ_{motor}), they increase the force and torque acting on the rigid body, resulting in higher accelerations ($\vec{a}$, $\vec{\alpha}$) and potentially higher angular velocities ($\vec{\omega}$) over time. This means that applying more motor effort translates into faster movement (higher accelerations) and larger force/torque outputs as experienced by the rigid body. The increased motion caused by greater motor effort leads to higher measured force and torque values as captured by an FT sensor mounted on the robot. The Newton-Euler equations show that $\vec{f}$ and $\vec{\tau}$ are directly influenced by $\vec{a}$, $\vec{\alpha}$, and $\vec{\omega}$.

When the same rigid body is moved with more effort (more torque from the motors), the result is greater accelerations and velocities. This, in turn, leads to higher force and torque measurements because:

$$\vec{f} \propto m\vec{a}, \quad \vec{\tau} \propto \mathbf{J}_s\vec{\alpha} \text{ and } \vec{\tau} \text{ also influenced by } [\vec{\omega}]^{\times}\mathbf{J}_s\vec{\omega}. \tag{10}$$

Hence, higher motor efforts produce stronger outputs in terms of $\vec{f}$ and $\vec{\tau}$, as the system's equations reflect this proportional increase.

The joint torques τ_{motor} drive joint motion, leading to increases in $\vec{a}$, $\vec{\alpha}$, and $\vec{\omega}$. The Newton-Euler equations use these motion variables to determine:

- $\vec{f}$ (force) scales with $m\vec{a}$.
- $\vec{\tau}$ (torque) scales with $\mathbf{J}_s\vec{\alpha}$ and includes gyroscopic effects from $[\vec{\omega}]^{\times}\mathbf{J}_s\vec{\omega}$.

Increased effort leads to higher $\vec{a}$, $\vec{\alpha}$, and $\vec{\omega}$, resulting in higher $\vec{f}$ and $\vec{\tau}$ (**??**).

2.2 Methods

The methods provide an overview of the approach used to isolate the force/torque (f/t) measurements arising from the rigid body ϕ_{gripper} from the actual f/t sensor measurements, thereby leaving only the measurements associated with ϕ_{payload}. This isolation allows for accurate analysis of the forces and torques attributable solely to the payload. At first the data preprocessing from the recorded sensor data to the train and test data is described (Sec. **??**). How the sensor data arises is seen in "Practical Implementation" (Sec. **??**). Subsequently the Gaussian Process Models Training and Testing is explained (Sec. **??**).

2.2.1 Data Preprocessing:

The data provided by the motor sensors of each joint motor and the force/torque-sensor are recorded and result in the Dataset:

$$\mathcal{D}_{\text{sensors}} = \left\{ \begin{array}{l} q^T \in \mathbb{R}^6,\ \dot{q}^T \in \mathbb{R}^6,\ \ddot{q}^T \in \mathbb{R}^6, \\ \boldsymbol{I}^T \in \mathbb{R}^6,\ \boldsymbol{f}^T \in \mathbb{R}^3,\ \boldsymbol{\tau}^T \in \mathbb{R}^3 \end{array} \right\} \in \mathbb{R}^{30} \tag{11}$$

where:

- q is the motors' positions,
- $\dot{q}$ is the motors' angular velocities,
- $\ddot{q}$ is the angular accelerations,
- $\boldsymbol{I}$ is the motors' effort,
- $\boldsymbol{f}$ is the force measured by the force/torque sensor (components x, y, z),
- $\boldsymbol{\tau}$ is the torque measured by the force/torque sensor (components x, y, z).

The data is then preprocessed to train the $\boldsymbol{GP}_{\text{effort}}$ and $\boldsymbol{GP}_{\text{wrench}}$ models, with slightly different preprocessing steps for each.

However the basis of the preprocessing process equals for both models. The input feature and output target vectors shape for both models remains the same:

$$\boldsymbol{x} \in \mathbb{R}^{18}, \boldsymbol{y} \in \mathbb{R}^6 \tag{12}$$

The Dataset $\mathcal{D}_{\text{sensors}}$ is reorganized for both $\boldsymbol{GP}_{\text{models}}$ into:

$$\mathcal{D}_{\text{raw}} \in \mathbb{R}^{mx24} \tag{13}$$

where m is the subsample size, $n = [0, 17]$ represents the input feature vector $\boldsymbol{x} \in \mathbb{R}^{18}$, and the output target vector $\boldsymbol{y} \in \mathbb{R}^6$ is represented by $n = [18, 23]$.

The $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{test}}$ Datasets for both models be:

$$\mathcal{D}_{\text{train}} \in \mathbb{R}^{mx24}, \mathcal{D}_{\text{test}} \in \mathbb{R}^{mx24} \tag{14}$$

and are processed as follows. $\mathcal{D}_{\text{raw}}$ includes the corresponding vectors of $\mathcal{D}_{\text{sensors}}$ to the input feature vector $\boldsymbol{x} \in \mathbb{R}^{18}$ and output feature $\boldsymbol{y} \in \mathbb{R}^{6}$ for both $\boldsymbol{GP}_{\text{models}}$.

The index m of $\mathcal{D}_{\text{raw}}$ is shuffled and $\mathcal{D}_{\text{raw}}$ is split into $\mathcal{D}_{\text{train}}$ reflecting 80% of $\mathcal{D}_{\text{raw}}$ and $\mathcal{D}_{\text{test}}$ reflecting 20% of $\mathcal{D}_{\text{raw}}$.

Now, $\mathcal{D}_{\text{train}}$ is standardized column-wise for all $n = 24$ columns to achieve a mean of 0 and a standard deviation of 1 for each column, using the 'StandardScaler' object from 'sklearn.preprocessing'. $\mathcal{D}_{\text{test}}$ is transformed with the $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ the 'StandardScaler' learned while transforming $\mathcal{D}_{\text{train}}$. This results in $\mathcal{D}_{\text{test}}$ having a mean close to 0 and a standard deviation close to 1 for each column. Standardizing in this way helps the model perform better when predicting on unseen data.

2.2.2 Gaussian Models:

The Gaussian Process model (GPM) is described by:

$$f(x) \sim \mathcal{GP}\left(m(x), k(x, x') + \sigma_n^2 I\right) \tag{15}$$

where $f(x)$ is the estimated target vector based on the sensor data. The features are represented by x, and the mean value function $m(x)$ describes the average relationship between x and $f(x)$, assumed to be zero because no prior information about the mean of the process is known. The covariance function $k(x, x')$ quantifies the correlation of the values of $f(x)$ at different points x and x', and in $\sigma_n^2 I$ the noise variance of the data is respected.

Both models are trained as Gaussian Sparse Regression models.

$$f(x) \approx \mathcal{GP}\left(m(x), k(x, \boldsymbol{Z})k(\boldsymbol{Z}, \boldsymbol{Z})^{-1}k(\boldsymbol{Z}, x) + \sigma_n^2 I\right) \tag{16}$$

where $f(x)$, x, $m(x)$, and $\sigma_n^2 I$ remain the same as in GP regression. The covariance function $k(x, \boldsymbol{Z})$ quantifies the correlation between the test point x and the inducing points $\boldsymbol{Z}$, while $k(\boldsymbol{Z}, \boldsymbol{Z})$ is the covariance matrix of the inducing points, and $k(\boldsymbol{Z}, x)$ represents the covariance between the inducing points and the test point. In sparse GP regression, the inducing points $\boldsymbol{Z}$ are selected as a subset of the training data or initialized randomly to cover the input space efficiently. These points are then optimized to capture the key structure and variability within the training data. After the optimization, the model is trained using these inducing points, which effectively summarize the broader dataset. This approach enables training on larger data volumes while maintaining computational feasibility and improves prediction accuracy by leveraging a broader representation of the data.

The RBF kernel $k_{\text{RBF}}(x, x')$ is used in the GPM, which is described by:

$$k_{\text{RBF}}(x, x') = \sigma^2 \exp\left(-\frac{\|x - x'\|^2}{2l^2}\right) \tag{17}$$

where x, x' represent the features, σ^2 is the variance of the kernel function, and l is the lengthscale parameter of the kernel. The RBF kernel is used because it allows an infinite number of turning points in the data and can be well adapted for multidimensional input spaces, which is necessary for four features.

A white noise kernel $[1]$ can be added to the RBF kernel to account for sensor noise. This kernel can be used during implementation on a real system to compensate for misbehavior of the GPM trained based on simulation data without sensor noise. Sensor noise can also be generated in the simulation, so the white noise kernel can also be implemented in the simulation if required.

$$k_{\text{white}}(x, x') = \sigma_n^2 \delta(x, x') \tag{18}$$

σ_n^2 is the noise in the white noise kernel itself, and the delta function δ ensures that the noise of the different features is uncorrelated.

To train the GPM optimally, the k-fold cross-validation training method is used [37]. The training dataset $\mathcal{D}_{\text{train}}$ (**??**) is divided into k equal parts $\{\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_k\}$. For each iteration i from 1 to k, one part $\mathcal{D}_i$ is used as the validation set, while the remaining $k-1$ parts $\mathcal{D} \setminus \mathcal{D}_i$ are used as the training set. The GPM is trained on the training set and evaluated on the validation set, and the process repeats k times so that each part of the dataset is used as a validation set exactly once. Performance metrics such as Mean Absolute Error (MAE), R^2, Mean Squared Error (MSE), and Root Mean Squared Error (RMSE) are recorded for each fold. Finally, the average of these metrics across all k folds is computed to assess the model's predictive accuracy and effectiveness.

- Mean Absolute Error (MAE): Average of the absolute differences between the predicted values and the actual values.
- R-squared (R^2): Measure of the goodness of fit, indicating how well unknown samples are likely predicted by the model. $R^2 \in [0, 1]$.
- Mean Squared Error (MSE): Average squared difference between the estimated values and the actual values.
- Root Mean Squared Error (RMSE): Measure of how accurately the model predicts the response, expressed in the same unit as the targets of the GPM.

GP effort model: A Sparse $\boldsymbol{GP}_{\text{effort}}$ (**??**) model is k-fold trained on its dataset $\mathcal{D}_{\text{train}}$ (**??**) with the input feature vector:

$$\boldsymbol{x} = \left\{q^T \in \mathbb{R}^6,\ \dot{q}^T \in \mathbb{R}^6,\ \ddot{q}^T \in \mathbb{R}^6\right\} \in \mathbb{R}^{18} \tag{19}$$

and the target vector:

$$\boldsymbol{y} = \left\{\boldsymbol{I}^T \in \mathbb{R}^6\right\} \in \mathbb{R}^6 \tag{20}$$

where:

$$\boldsymbol{x} = \left\{q_1^T,\ \dot{q}_1^T,\ \ddot{q}_1^T,\ \ldots,\ q_6^T,\ \dot{q}_6^T,\ \ddot{q}_6^T\right\} \in \mathbb{R}^{18} \tag{21}$$

and tested with $\mathcal{D}_{\text{test}}$ (**??**).

GP wrench model: A Sparse $\boldsymbol{GP}_{\text{wrench}}$ (**??**) model is k-fold trained on its dataset $\mathcal{D}_{\text{train}}$ (**??**) with the input feature vector:

$$\boldsymbol{x} = \left\{ q^T \in \mathbb{R}^6,\ \ddot{q}^T \in \mathbb{R}^6,\ \boldsymbol{I}^T \in \mathbb{R}^6 \right\} \in \mathbb{R}^{18} \tag{22}$$

and the target vector:

$$\boldsymbol{y} = \left\{ \boldsymbol{f}^T \in \mathbb{R}^3,\ \boldsymbol{\tau}^T \in \mathbb{R}^3 \right\} \in \mathbb{R}^6 \tag{23}$$

where:

$$\boldsymbol{x} = \left\{ q_1^T,\ \ddot{q}_1^T,\ \boldsymbol{I}_1^T,\ \ldots,\ q_6^T,\ \ddot{q}_6^T,\ \boldsymbol{I}_6^T \right\} \in \mathbb{R}^{18} \tag{24}$$

and tested with $\mathcal{D}_{\text{test}}$ (**??**).

Prediction with both GP models: To get the prediction that results in the f/t-measurements, that would appear when moving without the payload while carrying a payload, is done by both models. It is very important that the prediction of both models remain at the same timestamp. The $\boldsymbol{GP}_{\text{effort}}$ model predicts its target vector $\boldsymbol{y} \in \mathbb{R}^6$ (**??**)

then the input vector $\boldsymbol{x} \in \mathbb{R}^{18}$ (**??**) for the $\boldsymbol{GP}_{\text{wrench}}$ is processed where $q^T \in \mathbb{R}^6$ and $\ddot{q}^T \in \mathbb{R}^6$ are exactly the same features the $\boldsymbol{GP}_{\text{effort}}$ model predicted the $\boldsymbol{y} \in \mathbb{R}^6$ (**??**) with, as this combination is the input feature vector for the $\boldsymbol{GP}_{\text{effort}}$. The predicted target vector

$$\boldsymbol{y} = \left\{ \boldsymbol{f}^T \in \mathbb{R}^3,\ \boldsymbol{\tau}^T \in \mathbb{R}^3 \right\} \in \mathbb{R}^6$$

contains the f/t-measurement values that appear by the gripper while the robot executes its trajectory carrying the gripper and a payload.

3 Implementation

This approach, which isolates the external forces resulting from the gripper's rigid body ϕ_{gripper} through estimation, is implemented using a Robot Operating System (ROS)-based simulation [**ros_official**], with Gazebo [**gazebo_ros**] as the physics simulator and ROS MoveIt [**moveit**] as the motion planning interface for the UR5 robotic arm [**ur5_robot**]. The simulation setup leverages a `<xacro>` [**xacro_docs**] configuration to model the robot and its control interfaces and the gazebo simulation environment. The ROS Control framework [**ros_control**] is employed, integrating an effort transmission interface and an effort-based controller for controlling the UR5 robot in Gazebo. Additionally, a joint state interface is utilized to retrieve sensor feedback, including joint positions, velocities, and efforts. The system employs a joint_trajectory interface, enabling the effort joint_trajectory controller to provide comprehensive sensor data for the robot's motors encompassing position, velocity, and effort. The motor sensor data is achieved from the `joint_states` ROS topic [**rospy**]. The `<libgazebo_ros_ft_sensor.so>` plugin is used to generate the f/t-measurement with its measurement frame located between the force/torque-sensor flange and the gripper mounting plate. The f/t-measurement sensor data is achieved from the `/wrench` ROS topic [**rospy**]. The recorded sensor data is saved as $\mathcal{D}_{\text{sensors}}$ (**??**) and preprocessed for the $\boldsymbol{GP}_{\text{model}}$ training following (Sec. **??**). For handling file paths during this pipeline the Core Python `os` [**python_os**] library is used. The data preprocessing is done in Python 3 with the `pandas` [**pandas**] library for data manipulation and analysis, the `numpy` [**numpy**] library for numerical operations and array handling, and the `sklearn.preprocessing` [**scikit_learn**] library for scaling data. The $\boldsymbol{GP}_{\text{model}}$ training, following (Sec. **??**), is done with the `GPy` [**gpy**] library for Gaussian Process modeling, the `sklearn.model_selection` library for data splitting and cross-validation, and the `sklearn.metrics` library for evaluating model performance. The `numpy` library is also used for $\boldsymbol{GP}_{\text{model}}$ saving. For plotting graphs and visualizing results the `matplotlib.pyplot` [**matplotlib**] library is used. Two Datasets $\mathcal{D}$ (**??**) are recorded. Both contain Cartesian trajectories, whereby the sensor data of the robot is recorded during the execution of the trajectories. The data sets differ in the orientation of the TCP between the start and destination points of the recorded trajectories. For both data sets, the next target position is sampled random from a cube whose center is located on the TCP. The edge lengths of this cube are 1 meter. The maximum velocity and acceleration for the trajectories are sampled randomly from the value ranges $r_{\text{velocity}} = [0.01, 0.3]$ and $r_{\text{acceleration}} = [0.01, 0.5]$. The dataset $\mathcal{D}_{\text{Cartesian}}$ contains the sensor data of 1000 recorded Cartesian trajectories where the TCP and MF coordinate systems `xy`-planes are parallel to the environments `xy`-plane. The dataset $\mathcal{D}_{\text{CartesianOC}}$ (Orientation Change, OC) contains the sensor data of 1000 trajectories where the TCP's orientation is changed from the start point to the destination point of the trajectory. The trajectories were chosen because executing Cartesian paths without orientation changes reduces multivalence in the data compared to paths with orientation changes. The shaft position of one motor at two different timestamps can be the same, while the shaft positions of the other five motors differ at these two timestamps. This is why the same motor position at two different timestamps can require a different effort the motor uses, with respect to the dynamic inertia matrix $\boldsymbol{M}$ (**??**) of the robot. This approach enables the evaluation of how effectively a Gaussian process responds to multivariate distributions as an estimation method for payload assessment. For evaluation the trained $\boldsymbol{GP}_{\text{models}}$ 100 Cartesian trajectories are executed while the $\boldsymbol{GP}_{\text{models}}$ predict live. The models are live tested against the ground truth sensor measurements while executing trajectories computed in the same pattern then the trajectories the dataset contains with the models are trained with. All trajectories, both for training and during testing, are executed without payload. Therefore

$$\phi_{\text{effective}} = \phi_{\text{payload}} \tag{25}$$

what means that the difference between the actual f/t measurements and the estimated measurements is zero.

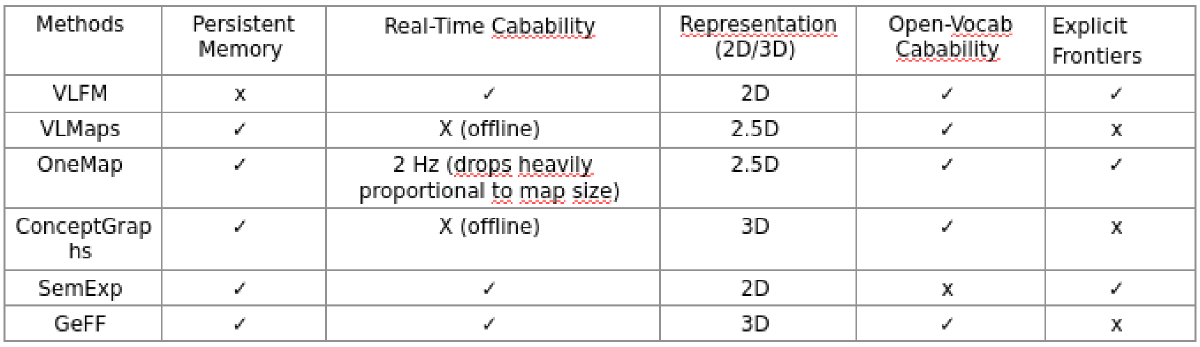
For each dataset, 10 $\boldsymbol{GP}_{\text{effort}}$ models and 10 $\boldsymbol{GP}_{\text{wrench}}$ models are trained sequentially with incrementally larger random sub samples. Starting with an initial subset defined by the step size, each model trains on a progressively larger portion of data. The number of inducing points is set to 3% of the sample size in each iteration, adapting dynamically to balance computational efficiency with model fidelity. All training loops are initialized with 70000 subsamples, this is why the first model is trained with 7000 subsamples and and the last model with 70000.

All models are tested predicting live while the robot executes its computed random trajectories. The $\boldsymbol{GP}_{\text{effort}}$ models are tested to evaluate the effort prediction against the motor effort ground truth, as well as the $\boldsymbol{GP}_{\text{wrench}}$ models to evaluate against the f/t measurement ground truth for both Dataset types $\mathcal{D}_{\text{Cartesian}}$ and $\mathcal{D}_{\text{CartesianOC}}$. In order to evaluate how accurately the resulting forces of the gripper can be isolated, the models that performed best for the respective data set are tested in series. For each Dataset $\mathcal{D}_{\text{Cartesian}}$ and $\mathcal{D}_{\text{CartesianOC}}$ one estimation against the ground truth (Sec. **??**)

$$\text{step size} = \frac{\text{total dataset size}}{10} \tag{26}$$

4 Discussion and Results

4.1 Results



Methods	Persistent Memory	Real-Time Cabability	Representation (2D/3D)	Open-Vocab Cabability	Explicit Frontiers
VLFM	x	✓	2D	✓	✓
VLMaps	✓	X (offline)	2.5D	✓	x
OneMap	✓	2 Hz (drops heavily proportional to map size)	2.5D	✓	✓
ConceptGraphs	✓	X (offline)	3D	✓	x
SemExp	✓	✓	2D	x	✓
GeFF	✓	✓	3D	✓	x

All $\boldsymbol{GP}_{\text{models}}$ are tested against the ground truth sensor measurement while live predicting. The results are structured in the results of the $\boldsymbol{GP}_{\text{effort}}$ models for both datasets $\mathcal{D}_{\text{Cartesian}}$ and $\mathcal{D}_{\text{CartesianOC}}$ and in the results of the $\boldsymbol{GP}_{\text{wrench}}$ for both datasets $\mathcal{D}_{\text{Cartesian}}$ and $\mathcal{D}_{\text{CartesianOC}}$. After that the serial prediction of both $\boldsymbol{GP}_{\text{models}}$ is estimation the resulting f/t measurements at the measurement frame. The estimation is the subtracted from the actual measurements to evaluate how accurate the $\boldsymbol{GP}_{\text{models}}$ isolate the f/t measurement resulting from the gripper. In this case the both best performing $\boldsymbol{GP}_{\text{models}}$ of each dataset $\mathcal{D}_{\text{Cartesian}}$ and $\mathcal{D}_{\text{CartesianOC}}$ are chosen and the isolation accuracy is evaluated for both trajectory patterns with and without orientation change.

4.1.1 Results GP_{effort} model

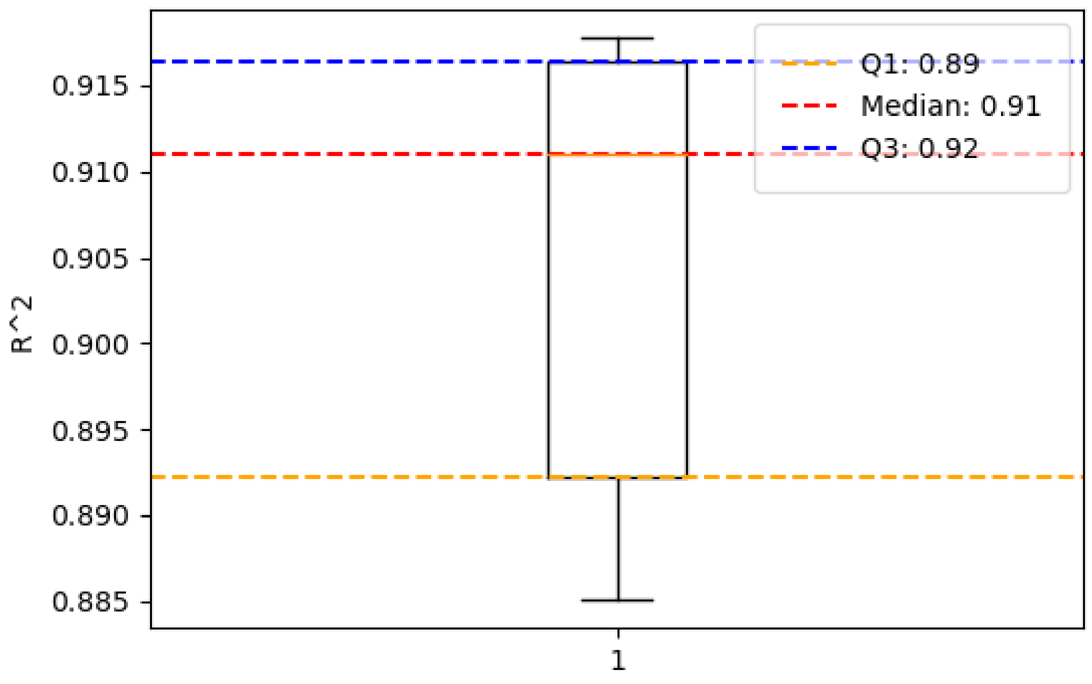


Figure 1: GP_{effort} model R^2 on effort $\mathcal{D}_{\text{Cartesian}}$ test data

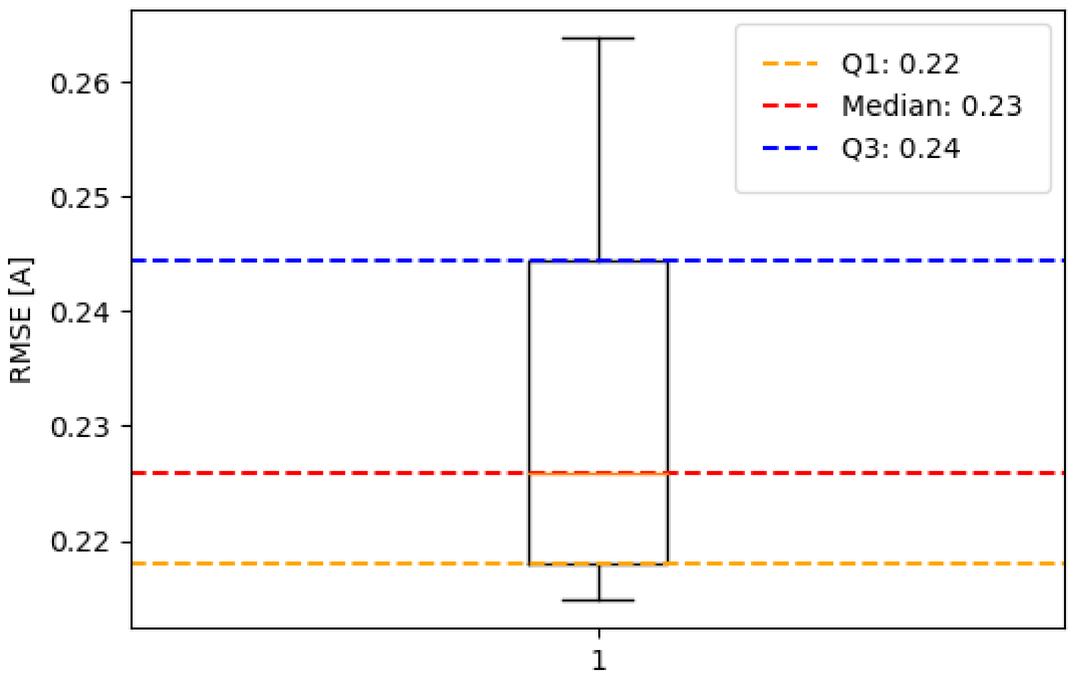


Figure 2: $\boldsymbol{GP}_{\text{effort}}$ model $RMSE$ on effort $\mathcal{D}_{\text{Cartesian}}$ test data

Model	**MAE**	**MSE**	**RMSE**	**R^2**
Model_1	0.1284	0.1207	0.2639	0.8851
Model_2	0.1186	0.1190	0.2556	0.8868
Model_3	0.1134	0.1196	0.2496	0.8863
Model_4	0.1141	0.0953	0.2286	0.9093
Model_5	0.1098	0.0964	0.2287	0.9083
Model_6	0.1080	0.0876	0.2179	0.9166
Model_7	0.1100	0.0918	0.2232	0.9127
Model_8	0.1071	0.0867	0.2154	0.9175
Model_9	0.1077	0.0885	0.2182	0.9158
Model_10	0.1070	0.0864	0.2148	0.9178

Table 1: $\boldsymbol{GP}_{\text{effort}}$ model metrics on effort $\mathcal{D}_{\text{Cartesian}}$ test data

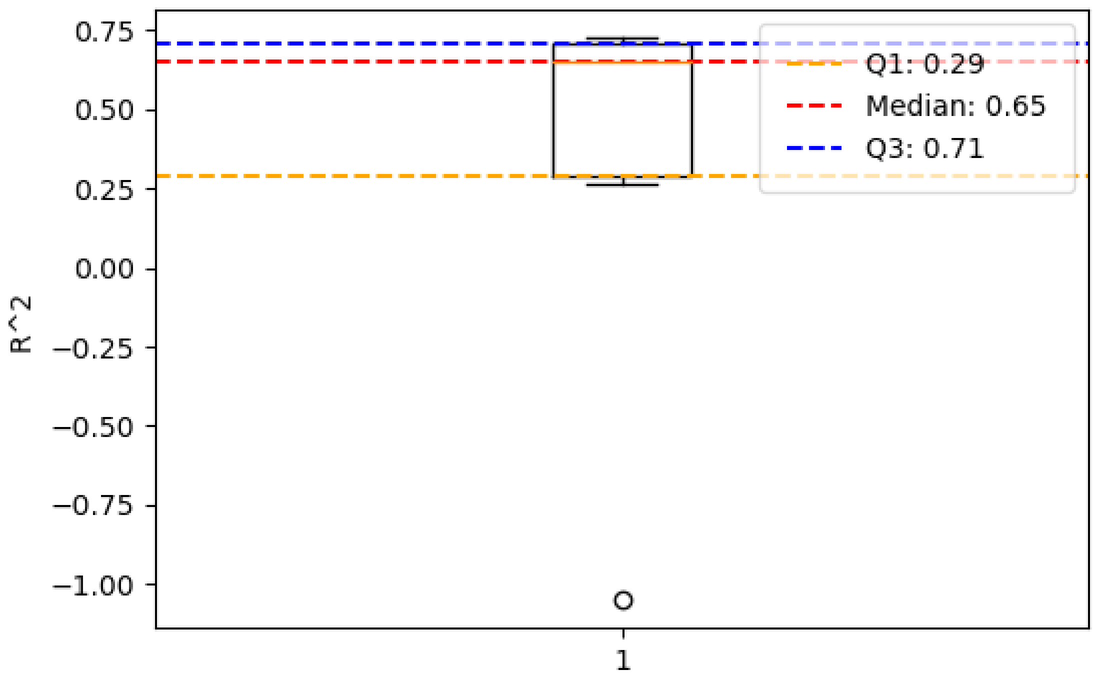


Figure 3: $\boldsymbol{GP}_{\text{effort}}$ model R^2 on effort Cartesian trajectories without orientation change live prediction

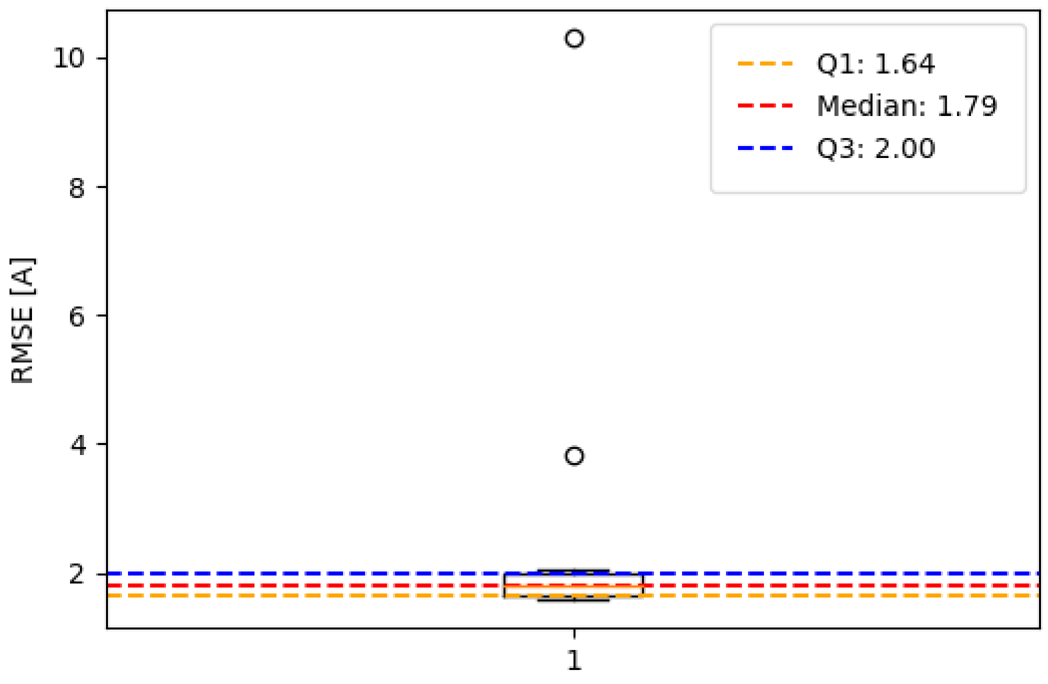


Figure 4: $\boldsymbol{GP}_{\text{effort}}$ model $RMSE$ on effort Cartesian trajectories without orientation change live prediction

Model	**MAE**	**MSE**	**RMSE**	**R^2**
Model_1	0.9905	3.8876	1.6011	0.7277
Model_2	1.0559	4.8169	1.7499	0.6838
Model_3	0.9799	4.2983	1.6435	0.6163
Model_4	7.0056	219.6050	10.3051	-1.0499
Model_5	1.0073	5.1650	1.8236	0.7022
Model_6	0.9912	3.7223	1.5833	0.7136
Model_7	1.0880	5.6405	1.8270	0.3171
Model_8	1.6480	31.9329	3.8415	0.2818
Model_9	0.9942	4.1817	1.6450	0.7131
Model_10	1.2684	7.4675	2.0570	0.2626

Table 2: $\boldsymbol{GP}_{\text{effort}}$ model live prediction metrics on effort Cartesian trajectories without changing orientation

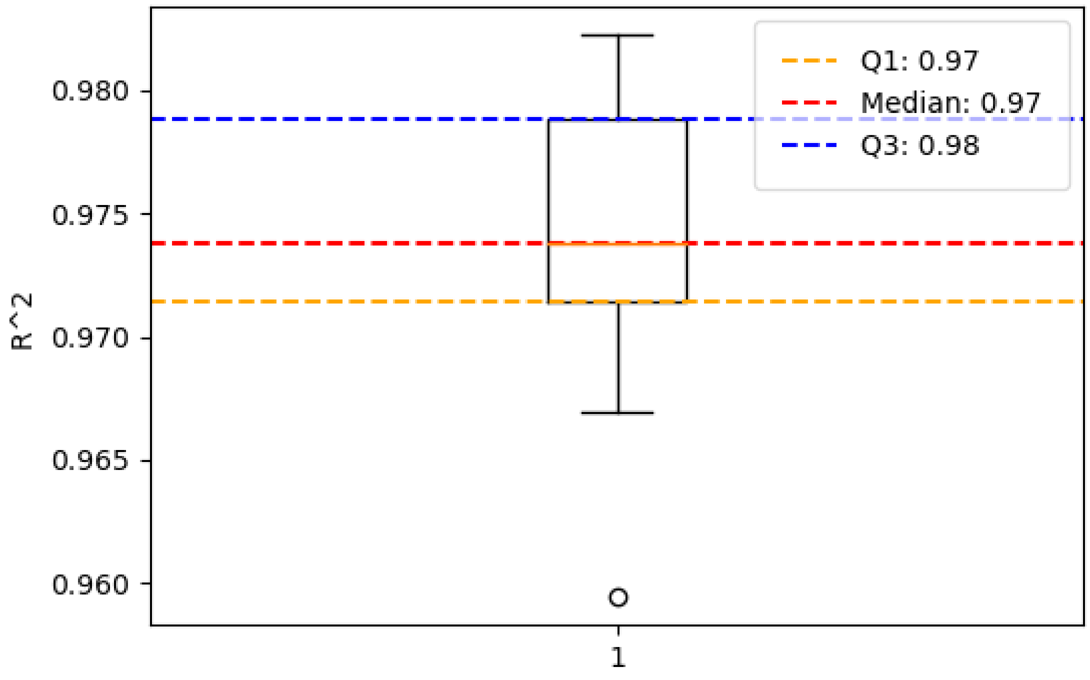


Figure 5: $\boldsymbol{GP}_{\text{effort}}$ model R^2 on effort $\mathcal{D}_{\text{CartesianOC}}$ test data

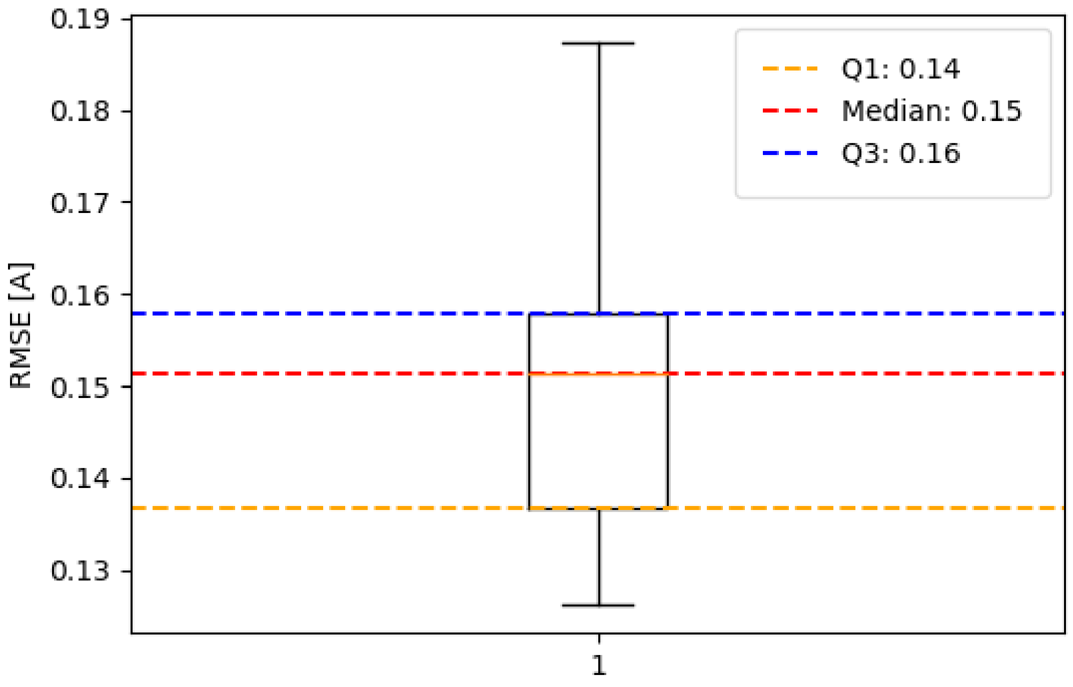


Figure 6: $\boldsymbol{GP}_{\text{effort}}$ model $RMSE$ on effort $\mathcal{D}_{\text{CartesianOC}}$ test data

Model	**MAE**	**MSE**	**RMSE**	**R^2**
Model_1	0.1156	0.0411	0.1873	0.9595
Model_2	0.1076	0.0335	0.1700	0.9670
Model_3	0.1034	0.0294	0.1590	0.9710
Model_4	0.1008	0.0278	0.1548	0.9726
Model_5	0.0942	0.0246	0.1460	0.9758
Model_6	0.0989	0.0267	0.1514	0.9737
Model_7	0.0984	0.0266	0.1511	0.9738
Model_8	0.0851	0.0204	0.1335	0.9799
Model_9	0.0802	0.0185	0.1277	0.9817
Model_10	0.0793	0.0180	0.1262	0.9822

Table 3: $\boldsymbol{GP}_{\text{effort}}$ model metrics on effort $\mathcal{D}_{\text{CartesianOC}}$ test data

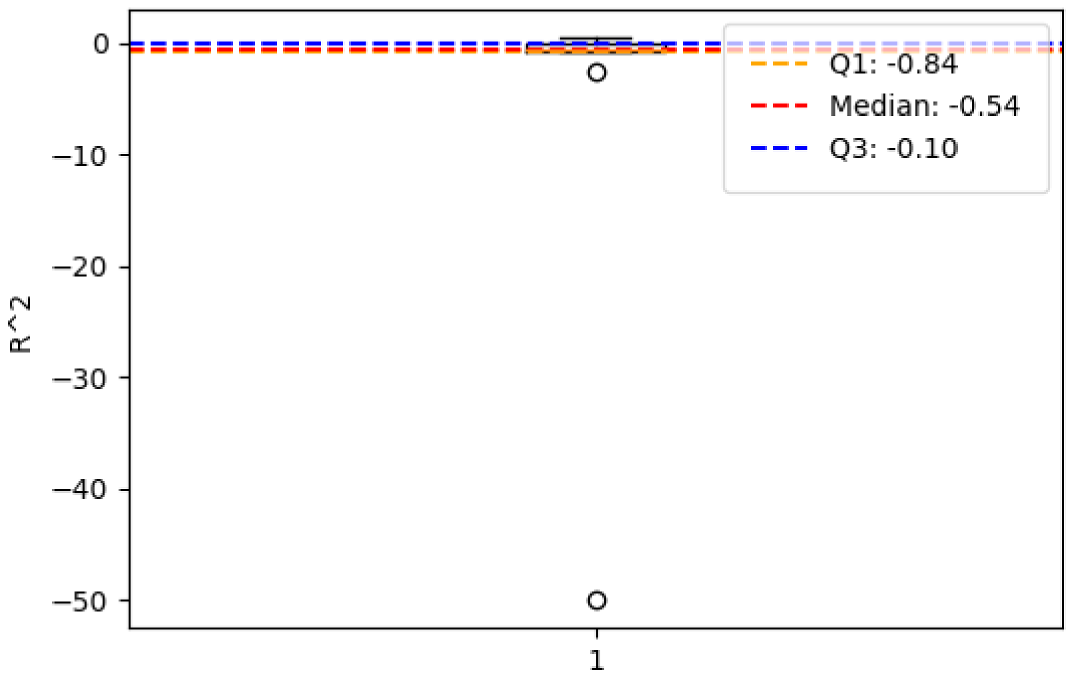


Figure 7: $\boldsymbol{GP}_{\text{effort}}$ model R^2 on effort Cartesian trajectories with orientation change live prediction

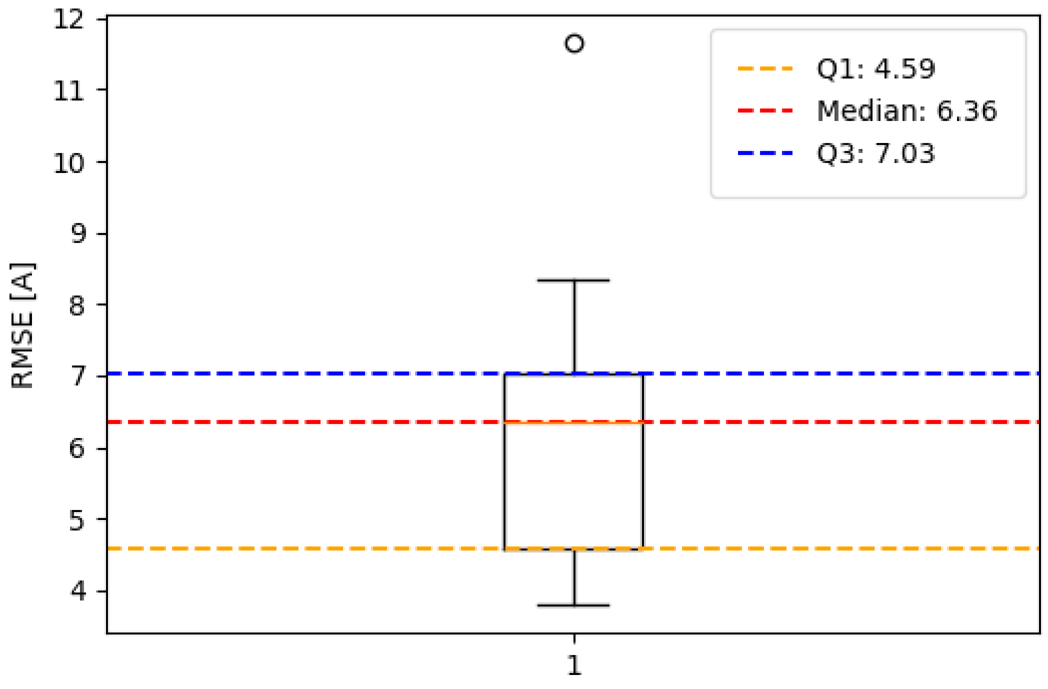


Figure 8: $\boldsymbol{GP}_{\text{effort}}$ model $RMSE$ on effort Cartesian trajectories with orientation change live prediction

Model	**MAE**	**MSE**	**RMSE**	**R^2**
Model_1	4.6183	44.5287	5.0202	-49.9557
Model_2	5.0939	84.8610	7.1296	-0.8461
Model_3	7.9057	170.7499	11.6557	-2.6656
Model_4	2.7165	21.7399	3.8019	-0.4782
Model_5	2.3954	22.5525	4.0519	0.4712
Model_6	3.7646	65.8977	6.6115	-0.0089
Model_7	3.8694	48.5134	6.1077	-0.3607
Model_8	5.1302	59.7537	6.7226	-0.8230
Model_9	2.7820	23.8206	4.4423	-0.0130
Model_10	5.9914	86.0272	8.3497	-0.6062

Table 4: $\boldsymbol{GP}_{\text{effort}}$ model live prediction metrics on effort Cartesian trajectories with changing orientation

4.1.2 Results $\boldsymbol{GP}_{\text{wrench}}$ model

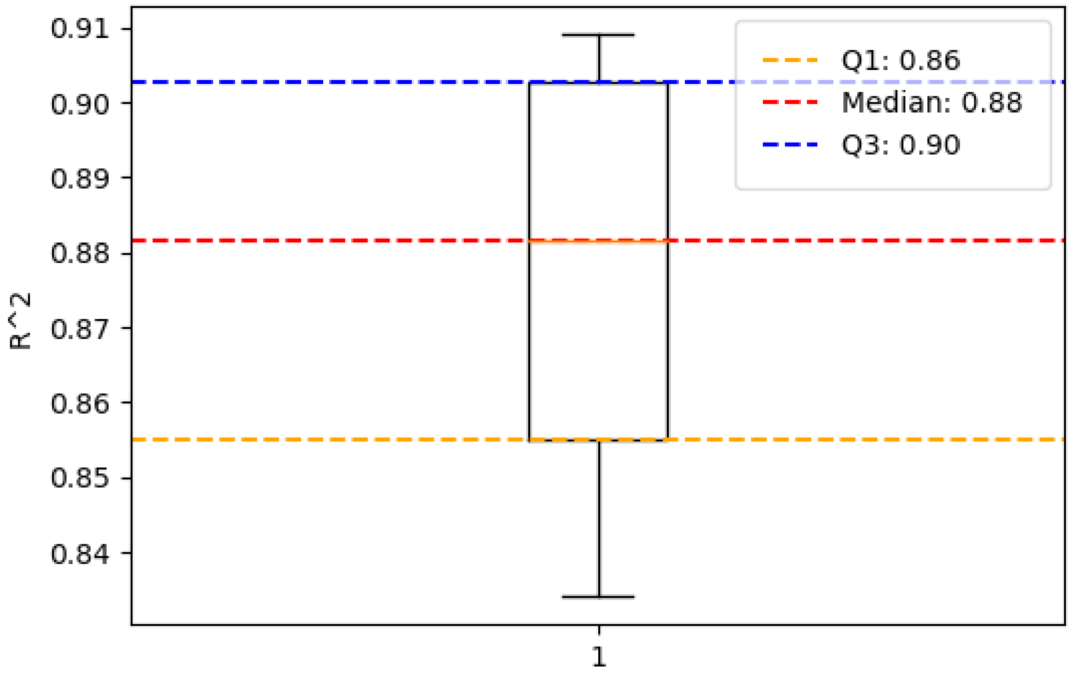


Figure 9: $\boldsymbol{GP}_{\text{wrench}}$ model R^2 on wrench $\mathcal{D}_{\text{Cartesian}}$ test data

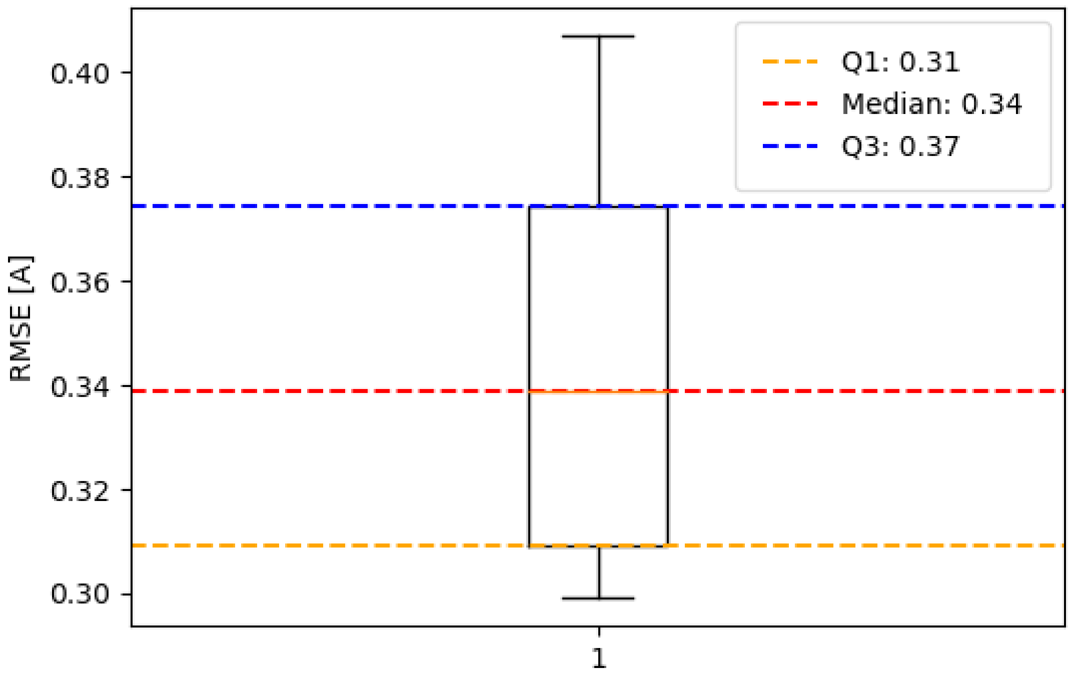


Figure 10: $\boldsymbol{GP}_{\text{wrench}}$ model $RMSE$ on wrench $\mathcal{D}_{\text{Cartesian}}$ test data

Model	**MAE**	**MSE**	**RMSE**	**R^2**
Model_1	0.1670	0.1724	0.4070	0.8341
Model_2	0.1441	0.1541	0.3778	0.8526
Model_3	0.1521	0.1661	0.3919	0.8410
Model_4	0.1458	0.1437	0.3636	0.8624
Model_5	0.1304	0.1118	0.3223	0.8930
Model_6	0.1264	0.0984	0.3047	0.9058
Model_7	0.1249	0.0949	0.2996	0.9091
Model_8	0.1254	0.0949	0.2992	0.9091
Model_9	0.1349	0.1141	0.3273	0.8907
Model_10	0.1361	0.1334	0.3502	0.8724

Table 5: $\boldsymbol{GP}_{\text{wrench}}$ model metrics on effort $\mathcal{D}_{\text{Cartesian}}$ test data

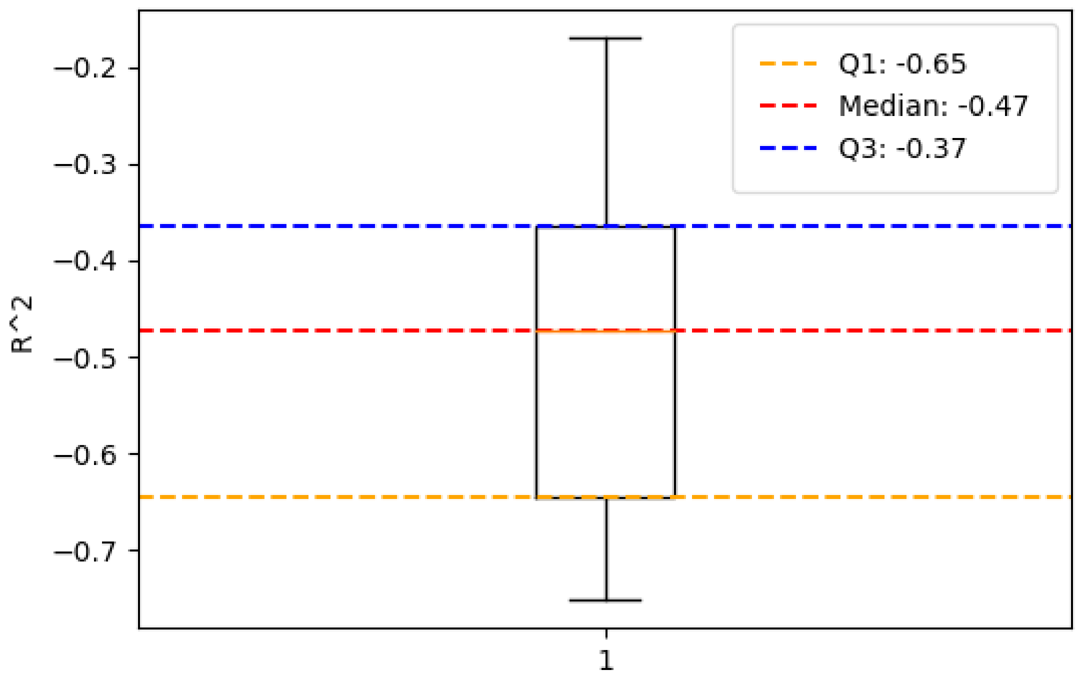


Figure 11: $\boldsymbol{GP}_{\text{wrench}}$ model R^2 on wrench Cartesian trajectories without orientation change live prediction

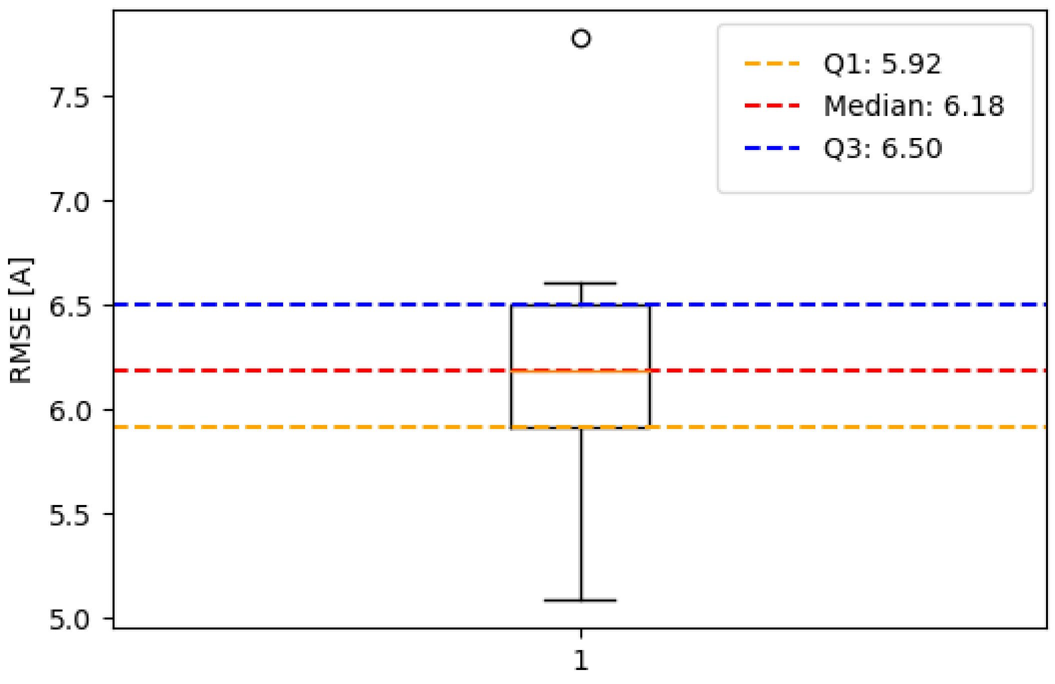


Figure 12: $\boldsymbol{GP}_{\text{wrench}}$ model $RMSE$ on wrench Cartesian trajectories without orientation change live prediction

Model	**MAE**	**MSE**	**RMSE**	**R^2**
Model_1	4.1511	67.9632	5.9142	-0.1697
Model_2	4.0221	78.1845	6.3824	-0.7511
Model_3	3.0775	49.3357	5.0893	-0.4293
Model_4	5.3184	115.9467	7.7806	-0.5945
Model_5	4.2552	79.7109	6.5438	-0.6628
Model_6	3.9019	68.8575	6.0438	-0.2886
Model_7	4.2663	74.2104	6.3233	-0.4425
Model_8	4.5354	81.6796	6.6045	-0.3442
Model_9	3.4870	61.9790	5.6544	-0.5044
Model_10	3.4664	68.1617	5.9179	-0.6679

Table 6: $\boldsymbol{GP}_{\text{effort}}$ model live prediction metrics on effort Cartesian trajectories without changing orientation

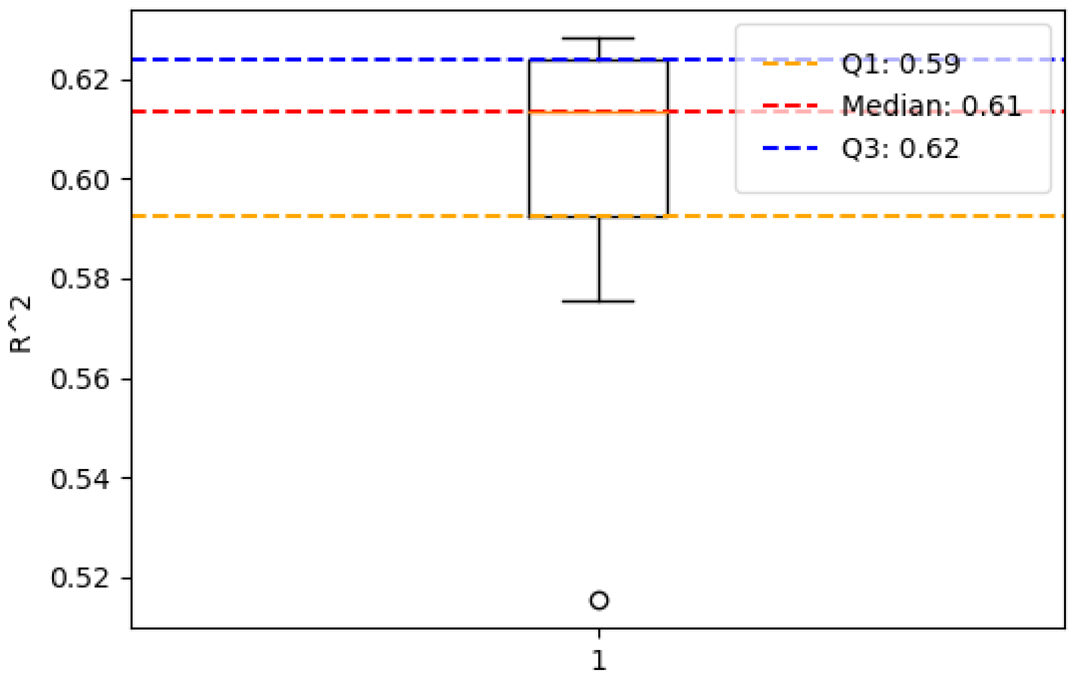


Figure 13: $\boldsymbol{GP}_{\text{wrench}}$ model R^2 on wrench $\mathcal{D}_{\text{CartesianOC}}$ test data

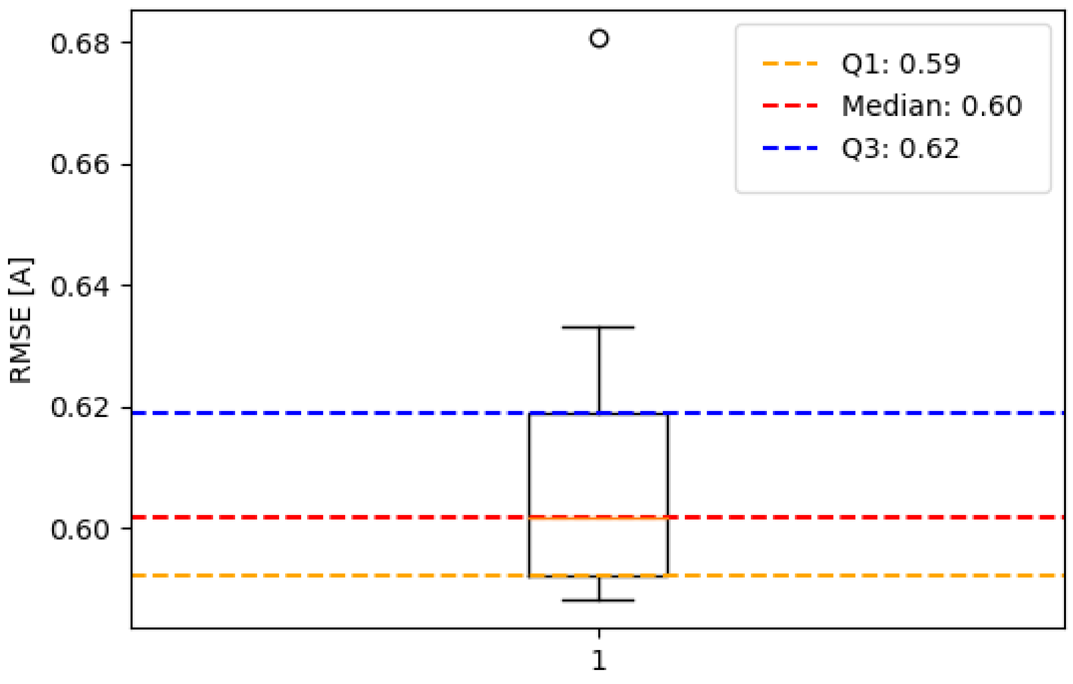


Figure 14: $\boldsymbol{GP}_{\text{wrench}}$ model $RMSE$ on wrench $\mathcal{D}_{\text{CartesianOC}}$ test data

Model	**MAE**	**MSE**	**RMSE**	**R^2**
Model_1	0.3196	0.4902	0.6807	0.5155
Model_2	0.2914	0.4290	0.6331	0.5757
Model_3	0.2798	0.4130	0.6202	0.5913
Model_4	0.2741	0.4087	0.6152	0.5957
Model_5	0.2688	0.3928	0.6038	0.6114
Model_6	0.2636	0.3890	0.5993	0.6153
Model_7	0.2588	0.3803	0.5925	0.6238
Model_8	0.2571	0.3801	0.5921	0.6240
Model_9	0.2541	0.3767	0.5893	0.6273
Model_10	0.2521	0.3757	0.5882	0.6283

Table 7: $\boldsymbol{GP}_{\text{wrench}}$ model metrics on effort $\mathcal{D}_{\text{CartesianOC}}$ test data

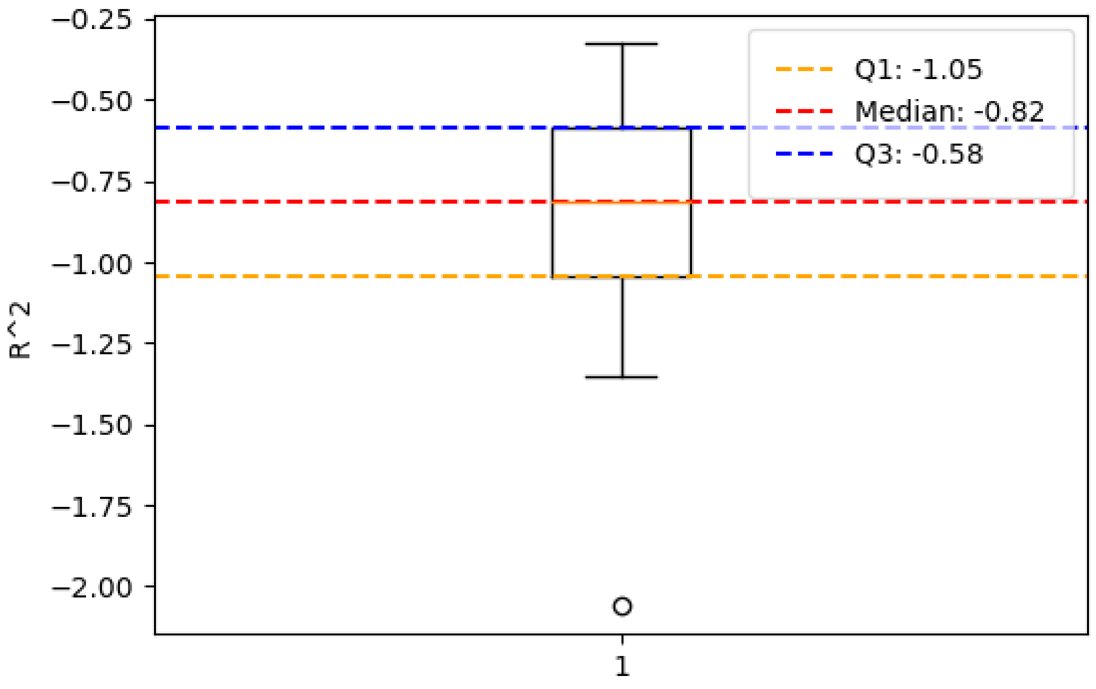


Figure 15: $\boldsymbol{GP}_{\text{wrench}}$ model R^2 on wrench Cartesian trajectories with orientation change live prediction

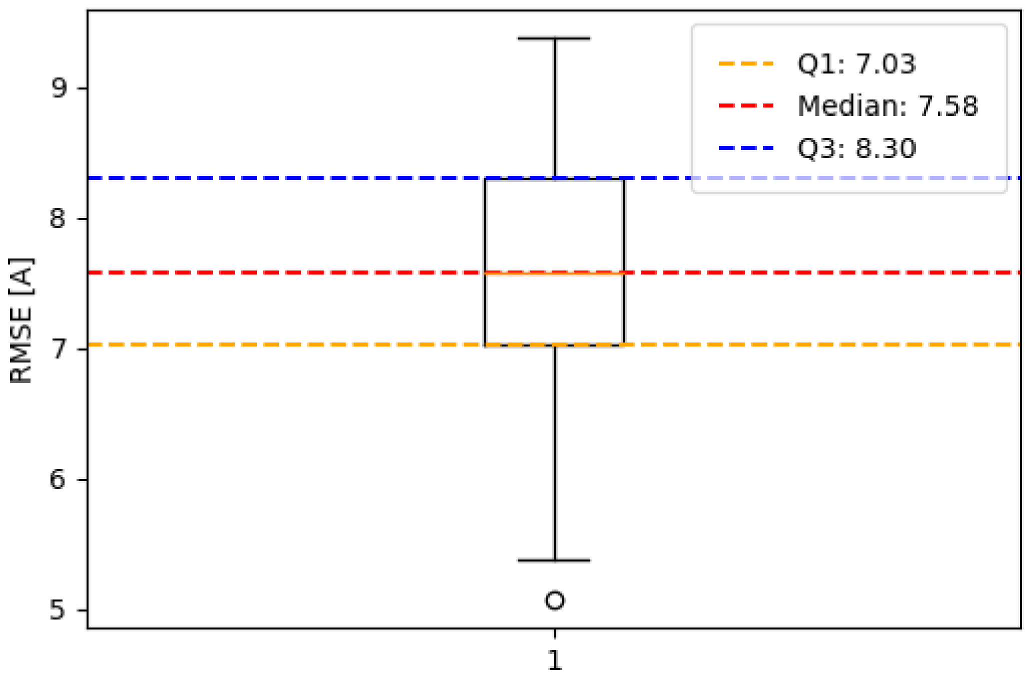


Figure 16: $\boldsymbol{GP}_{\text{wrench}}$ model $RMSE$ on wrench Cartesian trajectories with orientation change live prediction

Model	MAE	MSE	RMSE	R^2
Model_1	6.6921	153.0354	8.5885	-2.0593
Model_2	5.3542	111.5841	7.6753	-0.3246
Model_3	4.6114	110.5596	7.4602	-0.6125
Model_4	5.8744	138.9630	8.4449	-1.0565
Model_5	6.9017	179.3628	9.3817	-1.3500
Model_6	3.4584	48.1678	5.0710	-0.4637
Model_7	5.1164	107.0997	7.4934	-0.5723
Model_8	3.7980	53.8531	5.3796	-0.6411
Model_9	5.0121	88.8960	6.8829	-1.0151
Model_10	5.3321	125.7489	7.8648	-0.9897

Table 8: $\boldsymbol{GP}_{\text{wrench}}$ model live prediction metrics on wrench Cartesian trajectories with changing orientation

This chapter presents the experimental evaluation of the proposed hybrid semantic exploration system. Each experiment targets a specific research question and is evaluated using quantitative performance metrics.

4.2 Results on Semantic Multi-Object Search Tasks

4.2.1 Experiment 1: Single-Object Success Rate (SR)

4.2.2 Experiment 2: Navigation Efficiency (SPL)

4.2.3 Experiment 3: Multi-Object Success Rate (MSR)

4.2.4 Experiment 4: Ablation of Exploitation (OpenFusion)

4.3 Experiment 5: Improving Detection Robustness via Semantic Fusion

4.4 Experiment 6: Real-World Deployment

4.5 Experiment 7: Comparison of VL-Models for Frontier Scoring

5 Summary and Outlook

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

List of Figures

List of Tables

List of source codes

A Appendix A

B Appendix B