



**Universidad**  
**Zaragoza**

## Final Master Thesis

Robot grasping with Bayesian optimization informed  
with foundational models

Autor

Eduardo Pérez González

Director

Ruben Martínez Cantín

ESCUELA DE INGENIERÍA Y ARQUITECTURA  
2025



# AGRADECIMIENTOS

Firstly, I want to give my sincere thanks to the people who helped me to fulfill this Master's Thesis.

I am first and foremost immensely grateful to my supervisor, Rubén Martínez Cantín, for his guidance, patience, and unwavering support throughout the development of this project. His help in developing and addressing the challenges we've faced along the way has been invaluable.

I would also like to thank the team at the Institute of Engineering Research of Aragón (i3A) and their Robotics, Computer Vision, and Artificial Intelligence research group, whose collaboration and resources were crucial for making progress at different stages of this research. Broadly speaking, I want to note Gabriel, who explained context management in Bayesian Optimization. At the same time, I am deeply grateful to Javier García Barcos, my internship advisor along with Rubén, who has contributed with his guidance and support on closely related topics to the development of this thesis. The environment I am working in is very enriching, and working within it has allowed me to grow both academically and personally.

And finally, but certainly not least, I want to thank my family and friends, who have always provided support, understanding, and love through the toughest of times. Not to forget my younger brother, Daniel, whose unwavering belief in me has been my most significant motivation for reaching this milestone.

This work would not be possible without the support from all of these people. To each of them, I dedicate this project in deep appreciation.



# RESUMEN

Esta tesis de máster investiga técnicas avanzadas de manipulación robótica utilizando una mano robótica humanoide, combinando tecnologías de vanguardia en inteligencia artificial, como el aprendizaje autosupervisado (DINO) y la optimización bayesiana. El objetivo principal es desarrollar una canalización robusta y eficiente capaz de mejorar las estrategias de agarre mediante la integración de datos visuales y contextuales. La investigación se centra en aprovechar las características visuales preentrenadas extraídas por DINO y combinarlas con métricas espaciales dentro de un marco de optimización bayesiana, introduciendo un enfoque consciente del contexto que mejora la adaptabilidad y precisión en tareas de agarre.

Este trabajo se centra en desarrollar un proceso de entrenamiento que permita a los robots aprender a agarrar objetos de manera más precisa y eficiente. Para ello, se utiliza un enfoque que combina la extracción de información visual detallada, la evaluación de la calidad del agarre y la integración de estas características en un modelo avanzado de optimización. El método se probó en un entorno simulado con diferentes objetos, y los resultados mostraron que, a medida que el sistema va practicando con más objetos, mejora notablemente su capacidad de agarre.

Una de las claves del éxito es el uso de contexto: el sistema aprende de su entorno y utiliza esa información para adaptarse mejor a nuevas situaciones. Esto le permite generar configuraciones de agarre más estables y fiables. En las primeras etapas, cuando el modelo tiene pocos datos, su desempeño es limitado, pero conforme acumula más experiencia, se vuelve más robusto y eficiente. Esto es especialmente útil para tareas complejas, donde el sistema logra un rendimiento superior al utilizar el conocimiento del contexto.

En resumen, este trabajo combina técnicas de Optimización Bayesiana, aprendizaje auto-supervisado y conocimiento contextual previo para mejorar la capacidad de los robots en la manipulación de objetos. Los resultados son prometedores y marcan un avance importante hacia sistemas robóticos más inteligentes y capaces de trabajar en entornos dinámicos y desafiantes. Este estudio sienta las bases para futuros desarrollos en percepción robótica y manipulación autónoma.

# ABSTRACT

This Master’s thesis investigates advanced robotic manipulation techniques using a humanoid robotic hand, combining cutting-edge technologies in artificial intelligence such as self-supervised learning (DINO) and Bayesian Optimization. The primary objective is to develop a robust and efficient pipeline capable of enhancing grasping strategies through the integration of visual and contextual data. The research focuses on leveraging pre-trained visual features extracted by DINO and combining them with spatial metrics within a Bayesian Optimization framework, introducing a context-aware approach that improves adaptability and precision in grasping tasks.

This involves developing a training pipeline that can extract high-dimensional visual embeddings, generate grasp quality metrics, and integrate the aforementioned features into a deep kernel-based optimization model. The capabilities of the method are shown through experimentation on objects in a simulated, controlled setup. The system then increasingly optimizes for grasping by choosing objects in succession, showing that building contextual information allows the system to generalize to new scenes more effectively. Experiments indicate that the use of context significantly enhances convergence rate and optimizes the noise in predictions, ultimately making the grasping configurations far more reliable. Since the model has not accumulated enough data to adequately optimize early phases, optimization in the subsequent course of processing more objects leads to very apparent improvements in robustness and efficiency. In these cases, especially when the manipulation tasks are complex, the context-aware approach demonstrates better performance in stability and grasp quality.

This is a robotics work with a particular emphasis on combining Bayesian Optimization with self-supervised learning and prior contextual information for enhancing robot grasping. These results indicate an enabling prospect that there could be the leveraging of acquired knowledge by an intelligent system for efficient and fast object manipulation in dynamic and adverse environments. Here in this work, a primitive foundation is established, whereas in the future, significant enhancement in robotic perception with autonomous manipulation will be achieved.

# Index

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Objectives . . . . .	3
1.2	Methodology and Tools . . . . .	4
1.3	Thesis Structure . . . . .	5
<b>2</b>	<b>Theoretical Framework</b>	<b>7</b>
2.1	Principles of Robotic Grasping . . . . .	7
2.2	Grasping with Robotic Hands and the Role of Robotic Arms . . . . .	8
2.2.1	Grasping with Robotic Hands . . . . .	8
2.2.2	The Role of Robotic Arms in Grasping and Manipulation . . . . .	9
2.3	Grasp Quality Metric . . . . .	10
2.4	DINO: Self-Supervised Vision Transformer . . . . .	11
2.4.1	Training in DINO . . . . .	11
2.4.2	Inference in DINO for Robot Grasping . . . . .	12
2.5	Meta-Learning: Foundations of Adaptation . . . . .	13
2.6	Bayesian Optimization . . . . .	15
2.6.1	Gaussian Processes . . . . .	16
2.6.2	Acquisition functions . . . . .	17
2.6.3	Incorporating Context Variables in Bayesian Optimization . . . . .	19
2.7	Advanced Modeling: Deep Kernels and the ADFK-IFT Framework . . . . .	20
2.7.1	Fundamentals of Deep Kernels . . . . .	20
2.7.2	Approximate Deep Feature Kernels (ADFK) . . . . .	20
2.7.3	Independent Feature Transformations (IFT) . . . . .	21
<b>3</b>	<b>Project Pipeline</b>	<b>23</b>
3.1	Training Phase . . . . .	24
3.2	Context-Based Optimization Workflow . . . . .	24
<b>4</b>	<b>Setup</b>	<b>27</b>
4.1	Description of Objects . . . . .	27

4.1.1	Simulation Environment: Simox . . . . .	29
4.1.2	iCub Robot . . . . .	29
4.1.3	BayesOpt . . . . .	30
4.1.4	JAX for Deep Kernel Learning . . . . .	31
<b>5</b>	<b>Metodology</b>	<b>33</b>
5.1	Data Generation and Preprocessing . . . . .	33
5.1.1	Extraction of Visual Features with DINO . . . . .	33
5.1.2	Generation of Grasping Metrics and Logs . . . . .	34
5.1.3	Examples of Good and Poor Grasps . . . . .	35
5.1.4	Loading and Combining Data . . . . .	37
5.1.5	Data Splitting . . . . .	37
5.2	Deep Kernel Learning with JAX . . . . .	37
5.2.1	Input to JAX . . . . .	37
5.2.2	Metrics and Kernel in JAX . . . . .	38
5.2.3	Execution of Gaussian Process . . . . .	38
5.2.4	Incorporating Contextual Features . . . . .	38
5.3	Training Phases and Context Extraction . . . . .	39
5.3.1	Adaptation Phase . . . . .	39
5.3.2	Meta-Update Phase . . . . .	40
5.3.3	Context Extraction and Application . . . . .	40
5.4	Contextual Analysis . . . . .	40
5.4.1	Context-Aware Bayesian Optimization and Iterative Refinement	41
<b>6</b>	<b>Results: Impact of Context Integration</b>	<b>43</b>
6.1	Results Analysis and Observations . . . . .	43
6.2	Foundational Iterations: Establishing Context Through <i>boat</i> and <i>bottle</i>	47
6.3	Iteration 1: Analysis of Object Optimization . . . . .	48
6.4	Iteration 2: Analysis of Object Optimization (Reversed Order) . . . . .	51
6.5	Overall Discussion of the Results . . . . .	53
6.5.1	Future Directions and Practical Applications . . . . .	54
<b>7</b>	<b>Conclusions</b>	<b>55</b>
<b>8</b>	<b>Bibliography</b>	<b>57</b>
	<b>List of Figures</b>	<b>61</b>
	<b>List of Tables</b>	<b>63</b>



# Chapter 1

## Introduction

The dexterity of the human hand, its ability to grasp and manipulate objects of varying shapes, sizes, and textures—whether familiar or completely novel—is one of the defining features of human capability. This extraordinary skill has long inspired the field of robotics, where researchers aim to replicate human-like manipulation in robotic systems. Achieving this, however, is far from simple and presents numerous theoretical and practical challenges.

The crux of the issue has to do with how we grasp. Robust robotic grasping should take into account the shape and surface characteristics of the object, but also its weight and orientation, as well as its interaction with the environment. Humans, for example, note the difference and adjust accordingly with no major effort, while robots need to be extensively reprogrammed or recalibrated even for small variances in the objects they manipulate. This restricts their versatility, particularly in unstructured or dynamic environments, where a robot could potentially meet new objects that do not align with its pre-determined grasping methods.

Most existing robotic hands can be classified into one of two categories: either simple grippers or pseudo-dexterous, multi-fingered designs. After decades of cutting-edge research, the promise of a human-tempting hand-grasping ability has not yet been fulfilled. Uncertainties in properties such as weight distribution, friction, and deformability often make it difficult for robotic systems to model the anticipated motion of an object once it has been grasped. Finally, a solution to the best grasp configuration (the configuration of the robotic hand defined by the position, orientation, and posture of the robotic end effector) is computed through solving a sophisticated optimization problem that is computationally expensive and time-intensive.

To address these challenges, researchers are increasingly turning to advanced machine learning and optimization methods. In particular, **Bayesian Optimization (BO)** has emerged as a powerful approach for refining robotic grasping strategies. Unlike traditional trial-and-error techniques, BO uses probabilistic models to predict



Figure 1.1: Modern humanoid robotic hand showcasing advanced design for grasping and manipulation.

the success of potential grasps, refining its choices based on feedback from previous attempts. This iterative process enables robots to adapt more efficiently, learning to grasp diverse objects without requiring extensive reprogramming.

Meanwhile, techniques like **DINO (Distillation with No Labels)** have enabled the extraction of high-quality visual features from unlabeled data through self-supervised learning. These attributes are extremely useful to be given in classical robotic grasping, where an object 3D shape may be reasonably defined and these features can be used in combination with relevant spatial or contextual information leverage to inform final grasp decisions. This combination of these advanced techniques provides the opportunity to build systems which not just learn well but adapt to space and tasks.

The aim of this project is to improve humanoid robotic hands with a combination of **context-aware optimization**, **visual representation learning**, and **high degree of freedom robotic systems**. Building on such time-evolving contexts, the proposed method intends to enhance the accuracy, robustness, and workflow of robotic grasping by utilizing fantastic visual attributes that have been pre-trained and accumulated previously.

This work builds on recent advances in Bayesian optimization applied to robotics. This research expands on the latest developments in Bayesian optimization within the field of robotics. Previous studies, such as "Bayesian Multi-Solution Optimization for

Efficient Robotic Grasp Exploration,” have shown the promise of using probabilistic models to discover multiple grasp configurations for unfamiliar objects, thereby improving the efficiency of robotic manipulation tasks [1].

## 1.1 Objectives

This work employs a pretrained frozen DINO (Distillation with NoLabels) model as part of a robotic grasping pipeline, and optimizes its performance using Gaussian Process Bayesian Optimization (BO). Grasping strategies are simplified and enriched with DINO-relearned visual features without the requirement for additional contextual integration. The method follows approaches as in Caron et al. (2022) [2], which is more concrete in the use of self-supervised visual transformers for feature extraction, employing these features in robotic grasping tasks. The goal here is to ingest and verify its success on the objects that had been included in the training dataset and test whether our grasp performance is reliable and comparable to the training conditions.

- **Integrating DINO into the Grasping Pipeline:** Incorporate a frozen DINO model into the robotic grasping pipeline with the aim of extract robust visual features. In the process, these extracted features work seamlessly with Bayesian optimization.
- **Designing an Efficient Optimization Process:** Design a lightweight and efficient pipeline that leverages DINO’s visual embeddings to refine, i.e., score, the optimization of robotic grasps.
- **Validation with Training Objects:** Validate the pipeline with data only seen by the training dataset. which will allow the system to have the ability to reproduce successful grasps in familiar situations.
- **Simulation and Experimental Testing:** Use the Simox robotic simulator to create and assess the pipeline. Modeling, planning and evaluation of grasp quality on predetermined targets in a known and controlled environment.
- **Baseline Comparison:** This pipeline is also compared with baseline version without DINO, Evaluating the effects of DINO’s visual embeddings on the final grasp success can be aided through this.

## 1.2 Methodology and Tools

The methods proposed in this paper will be realized through an integration of higher level machine learning tools along with simulation environments designed for manipulation tasks. One major component of this process is using Bayesian optimization to improve grasping methods. In this section, we explore GPE as an optimization tool, which is a probabilistic approach to optimizing complex, computationally intensive and non-transparent black-box functions—ideal for fine-tuning grasp configurations. Additionally, this implementation is using very modern libraries like JAX, for high-performance computation, to allow development of custom Gaussian Process kernels including visual and spatial features for grasping-type tasks.

The virtualization of these manipulation and optimization experiments will be realized through a simulation environment with the help of the Simox library. Simox is a lightweight, platform neutral simulator to implement robotic tasks like grasp planning and evaluation in an efficient manner. It is built on three key libraries: VirtualRobot for robot model and robot descriptions, Saba for motion planning, and GraspStudio for evaluating grasp quality. Overall, these capabilities make for an environment that allows fine-grained experimentation, while being more computationally efficient than larger simulation frameworks such as ROS. Flexibility of Simox supports easy integration by adding new experimental setups and configurations. The experimental workflow described in this study provides a new approach to composing the culture of the customized experiment execution system. It aims to facilitate the drop-in adoption of new optimization strategies, ablations, and assessment metrics. This framework is developed in python3 to build systems for rapid prototyping and easy experiment management. This framework makes it easy to keep track of results, enabling quick comparisons between optimization strategies and contexts. All together, these tools and methodologies offer a solid and flexible framework for effectively testing and optimizing robotic grasping strategies.

A Gantt chart (Figure 1.2) was created to help manage the project in a systematic manner. The timeline breaks the work into five main tasks over the course of several months. The first task will be to perform an extensive literature review on foundational models, computer vision, and robotic grasping. This initial phase is critical for establishing a solid theoretical foundation and will entail examinations of the utilization of Bayesian optimization in robotics; the employment of DINO for visual feature extraction; and the deployment of meta-learning methodologies to robotic platforms.

Afterward, DINO will be integrated with the simulation environment Simox,

allowing the creation of robotic grasping simulations and elaborate logging. The logs will combine visual s features and grasping metrics that the next training and optimization stages will require. The next step is to implement and improve the model pipeline, using JAX to optimize through Bayesian optimization and Gaussian Process modeling and to make continuous updates to the grasping policy through ongoing evaluations on varied objects.

In the final steps, this includes a qualitative and qualitative evaluation, as well further refinement of the model. This phase will close any remaining performance gaps and tune the grasping strategies. Finally, a documentation phase will round up all results, analysis, conclusions, making results available and clear for further development.

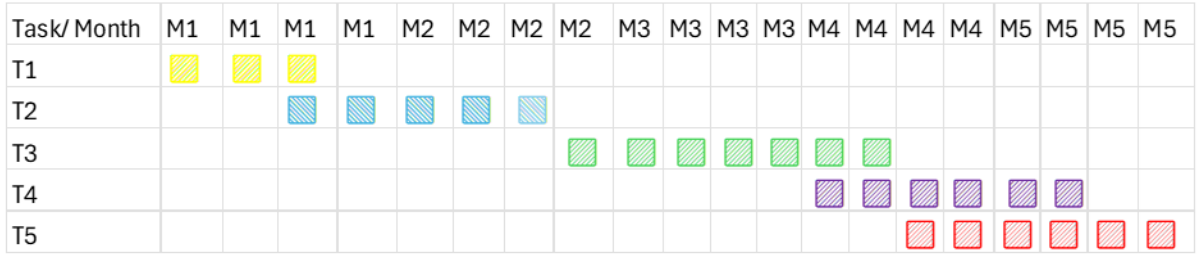


Figure 1.2: Gantt chart illustrating the timeline and structure of the project, showing the distribution of tasks across several months.

This structured approach, supported by the Gantt chart, ensures a well-defined timeline and facilitates the seamless progression from theoretical research to practical implementation, optimization, and documentation. It provides a clear roadmap for completing the project efficiently while maintaining a high standard of quality.

## 1.3 Thesis Structure

This thesis is organized to guide readers through the theoretical foundations, methodological details, and practical results of the research. Chapter 2 discusses the principles of robotic grasping and the integration of Bayesian Optimization with contextual modeling. It also introduces the DINO framework for self-supervised learning and its application to extracting robust visual features. Chapter 3 outlines the project pipeline, describing the training process that aligns visual and spatial data for enhanced grasp optimization and detailing the integration of Bayesian Optimization and Gaussian Processes. Chapter 4 presents the experimental setup, including the simulation environment and the iCub humanoid robot, alongside an overview of the selected object dataset. Chapter 5 explains the methodology, detailing the data generation, preprocessing, and optimization strategies used to refine grasp

configurations. Chapter 6 showcases the results, emphasizing the role of context-aware approaches in improving grasp quality and analyzing performance across different objects. Finally, Chapter 7 concludes with the key findings, their broader implications, and future directions for extending the methodologies to more complex scenarios and dynamic environments. This structure ensures clarity in presenting both the technical details and the significant contributions of the research.

# Chapter 2

## Theoretical Framework

### 2.1 Principles of Robotic Grasping

In simple terms, a robotic grasp is the way a robot hand or end-effector holds and secures an object so that it does not slip or fall. This concept is important in many robotic applications—such as pick-and-place tasks in factories—because it ensures that the robot can move or manipulate the object safely and reliably.

When a robot grasps an object, it usually forms a set of discrete points of contact between the fingers (or gripping surfaces) and the object. How it grasps and how strong of a grip it has on things relies on several things. The relative position and orientation of the hand to the object is thus important because closely aligning the robot’s gripper with the object can lead to stable or unstable grasps [3]. Additionally, the contact forces at each fingertip, particularly friction, play a critical role in securely holding the object during movement or manipulation [4]. Finally, the geometry and mobility of the robot’s hand or gripper influence its ability to conform to the shape and size of the object, as well as the effectiveness and precision of complex maneuvers.

A good grasp makes it easier for the robot to lift the object, move it around, and perform further actions (like rotating or inserting it into a container). Conversely, a poor grasp might lead to slips, drops, or damage. Therefore, in robotic manipulation, one of the main goals is to find and maintain a grasp configuration that is strong and stable enough for the task at hand, taking into account both the physical properties of the object (size, shape, weight, etc.) and the robot’s mechanical limitations.

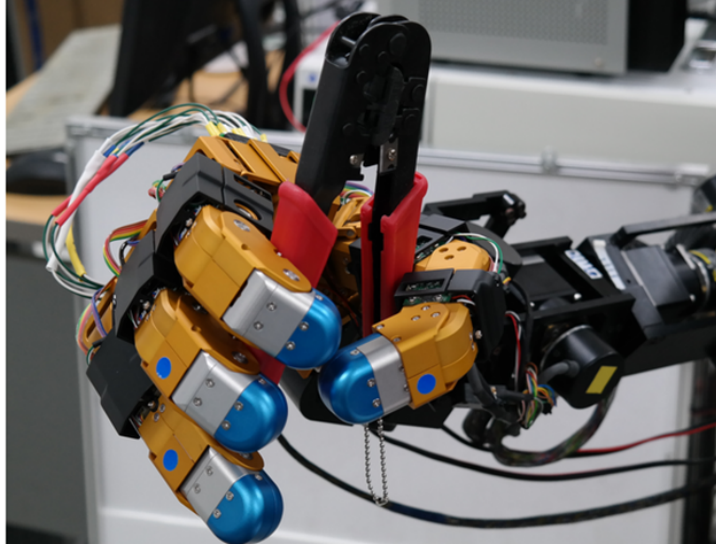


Figure 2.1: An example of a robotic hand performing a grasp on an object. The image highlights the contact points and alignment between the hand and the object.

## 2.2 Grasping with Robotic Hands and the Role of Robotic Arms

### 2.2.1 Grasping with Robotic Hands

Stable and effective grasps are crucial for successful manipulation tasks with dexterous robotic hands. When the target object is novel, however, reasoning about the shape and properties of the object in different configurations and conditions is necessary to produce the optimal grasp. Specialised sensing technologies, such as LiDAR, infrared imaging and haptic feedback systems, have been used to deliver detailed information on the geometry and material characteristics of an object. Nevertheless, this work is concerned with fully exploiting the capabilities of the tactile sensors mounted inside the robotic hand, avoiding the necessity of having external-use sensors or complicated machine learning models demanding too much resources.

The primary goal is to maximize a grasp quality metric  $Q$ , derived solely from tactile feedback, to determine the optimal grasping configuration. Formally, this problem can be expressed as:

$$w^* = \operatorname{argmax}_{w \in W} Q(w),$$

where  $w$  is the Tool Center Point (TCP) pose, and  $W$  represents the workspace reachable by the robotic hand. The TCP pose is described in Cartesian coordinates  $(X, Y, Z)$  for position and roll, pitch, and yaw angles for orientation, such that  $w \in W \subset \mathbb{R}^6$ . [3]



In addition to TCP pose, the configuration of the robotic hand’s intrinsic degrees of freedom (DoF) significantly influences its ability to perform effective grasps. These DoFs allow the hand to adjust its posture to match the geometry and physical characteristics of the object. When combining both TCP pose  $w$  and hand posture  $p$ , the optimization problem expands to:

$$w^*, p^* = \operatorname{argmax}_{w \in W, p \in P} Q(w, p), \quad P \subset \mathbb{R}^d,$$

where  $P$  represents the space of feasible hand postures, and  $d$  corresponds to the number of DoFs in the robotic hand. Optimizing the hand posture can be computationally expensive due to its high dimensionality [5]. To simplify the problem, this work focuses solely on optimizing the TCP pose, while the concept of posture synergies—dimensionality-reduced subspaces that retain effective hand configurations—is acknowledged but not directly implemented due to the lack of publicly available data.

Furthermore, this work stresses the importance of sampling multiple grasp configurations from the varying areas of an object. Things which have varying spatial properties but have no functional use separate these regions. Some possible grasp areas of a coffee mug could be: handle, rim, or the body. As an example, we show the various robotic grasp configurations in robotic systems as shown in Figure 2.5.

To simplify the problem, only single-handed grasps are considered in this work.

### 2.2.2 The Role of Robotic Arms in Grasping and Manipulation

The arms provide the precision required to place and orient the end-effector (hand) or tool in a three-dimensional space. Consisting of several degrees of freedom, robotic arms allow for intricate movements enabling the reach of desired objects in confined spaces, adjusting the angle of approach and positioning of an object for a particular task [6]. This ability allows the robotic hand to assess a multitude of items and create grasps according to an object’s composition [7].

Robotic arms also play a critical role in addressing environmental constraints. Factors such as the object’s orientation, the presence of obstacles, or the need to apply specific forces during manipulation are managed through the arm’s coordinated movements. For example, the arm can align the TCP to approach the object from an optimal direction, ensuring stable and effective grasp configurations even in cluttered or dynamic environments [8].

In the context of this work, the ability of the robotic arm to position the TCP within
















		Power		Precision	
		Palm		Pad	
Thumb adducted					
	Large Diameter	Medium Wrap	Ring	Palmar Pinch	Tripod
Thumb abducted					
	Small Diameter	Power Sphere	Sphere 3 Finger	Tip Pinch	Inferior Pincer
Thumb adducted					
	Adducted Thumb	Fixed Hook		Prismatic 2 Finger	
Thumb abducted					
	Light Tool			Parallel Extension	

Figure 2.2: The classification of robotic grasps: 15 types of commonly used grasping modes. This diagram highlights the diversity of grasping strategies that robotic hands can adopt.

the reachable workspace is vital for adapting to different objects and environmental conditions. This adaptability not only improves the quality of the grasp but also reduces reliance on external sensors, creating a more streamlined and efficient system. By integrating the tactile feedback from the robotic hand with the arm’s spatial flexibility, the overall system achieves higher levels of autonomy and robustness.

The collaboration between robotic arms and hands enhances the precision and efficiency of grasping systems. The arm provides the spatial control needed for positioning, while the hand delivers detailed tactile feedback for fine-tuning the grasp. Together, they enable sophisticated manipulation capabilities that are crucial for performing complex tasks in dynamic and unpredictable environments.

## 2.3 Grasp Quality Metric

The grasp quality is evaluated using the epsilon metric [9], which measures the radius,  $\epsilon$ , of the largest 6D ball centered at the origin enclosed in the Grasp Wrench Space (GWS). This metric assumes constraints in the finger forces, where the total force  $f$  satisfies:

$$f = \sum_{i=1}^n \sum_{j=1}^m \alpha_{ij} f_{ij}, \quad \alpha_{ij} \geq 0, \quad \sum_{i=1}^n \sum_{j=1}^m \alpha_{ij} \leq 1.$$

The total wrench  $\omega$  is similarly computed, and the GWS is defined by the convex hull of all possible  $\omega$ . A variation of this metric adds a Collision Penalty (CP) [10] to improve optimization convergence:

$$Q = Q_{\epsilon} + CP, \quad CP = 1 - e^{-\lambda n}.$$

Although the CP variation is not used in this work, it is included for potential future research.

## 2.4 DINO: Self-Supervised Vision Transformer

DINO (Distillation with No Labels) is a self-supervised learning approach for ViTs which allows model training without having access to labeled datasets. This is particularly beneficial for robot grasping, since DINO can extract high-quality visual features from unlabeled images. For example, during a common grasping task, the robot receives image input of the object to be grasped, figures out features of the object and what to do with it, with tasks like grasping and manipulation. DINO, most notably DINOv2 [11], is a resilient framework for producing the feature maps required for object detection and manipulation. The DINO method itself indeed covers a specific training process, but to avoid confusion, it is not used the DINO method for the purpose of training in this work. In this paper, it is only used a pre-trained DINO model in the inference stage and do not re-train it, and utilize its pre-learned visual representations in the context of robotic grasping tasks.

### 2.4.1 Training in DINO

DINO (Distillation with No Labels) is trained in a self-supervised manner, meaning it learns meaningful visual representations without manually labeled data. It uses a Teacher-Student architecture, where both models are ViTs (image classification portion of ViT) [11]. More specifically, the teacher model is maintained by an EMA of the student model parameters, leading to increasingly rich and semantic representations. Such mechanism is especially useful in tasks like robotic grasping as the model effectively learns about the world with no explicit supervision regarding the object.

DINO also utilizes a self-distillation strategy, where the student model learns to replicate the outputs of the teacher model. This eliminates the need for labeled data, significantly reducing the cost and time required for data preparation. It also allows the

system to understand essential object features such as shape, size, and texture directly from images. Finally, the multi-crop strategy enhances the learning of consistent representations by generating multiple augmented views of the same image. These views include both wide (“global”) and small (“local”) crops, improving the model’s ability to handle variations in size and perspective—an essential aspect for object grasping in real-world scenarios.

DINO is computationally expensive, but, can be obtained a powerful visual representation that generalizes so well across contexts and situations that we do not need to retrain it.

### 2.4.2 Inference in DINO for Robot Grasping

This paper considers only the inference stage of DINO, using a pretrained or “frozen” model. In inference time, DINO would obtain robust and generalized visual representations from the object images efficiently. These enable the robot to calculate best grasping positions and understand the shape [11]. Such visual analysis is important to plan activities with objects, like their accurate and efficient manipulation.

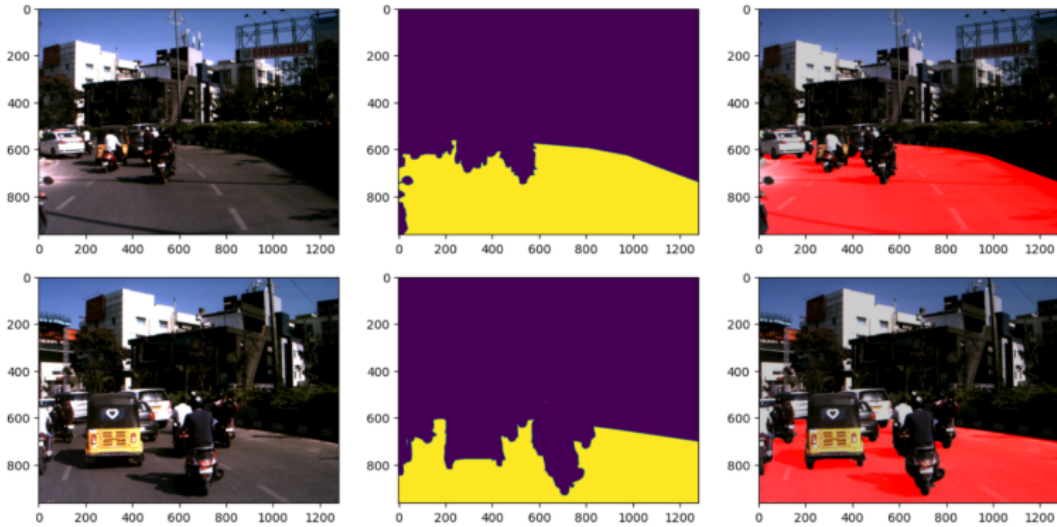


Figure 2.3: Visualization of DINOv2’s capabilities: The framework processes images (left) to produce the segmentation maps (middle) and to apply visual overlays (right). These illustrate how high-quality representations learned in DINO can make robust, spatially consistent features to solve downstream tasks, such as grasp prediction in robotic systems. Adapted from [11].

DINO’s capabilities also enhance the robot’s adaptability to varying visual conditions and unfamiliar objects. By leveraging the learned representations, the robot can perform tasks such as grasp prediction, object manipulation in different environments, and visual-based interaction with the world. These abilities not only

improve the robot’s performance in simple tasks but also prepare it for more complex activities like sorting, assembling, or reorganizing objects in dynamic scenarios.

One of the significant advantages of frozen DINO is its ability to eliminate the computational and time costs associated with training [11]. The extracted visual features are sufficiently comprehensive to deliver stable and adaptable performance, even in previously unseen scenarios. This blend of real-time efficiency and flexibility makes DINO a valuable tool for robotic grasping tasks, where systems must quickly analyze visual data and make precise decisions in complex environments.

DINOv2 represents a major evolution in what is possible in visual self-supervised learning and from it shows potential to teach robots to understand object features independently and possess a greater understanding of their physical interactions with the world.

## 2.5 Meta-Learning: Foundations of Adaptation

Meta-Learning, or "learning to learn," is a machine learning approach that allows models to adapt quickly to new tasks with only a small amount of data. Meta-Learning, in contrast to classical methods optimized for performance in a single task, rather focuses on providing task as well as architecture specific algorithms that are capable of quickly learning from a few examples and generalizing well to new tasks [12]. It uses knowledge gained from previous tasks] to build a base that allows solving the next task with a small amount of computation [13].

In theory, Meta-Learning is a second layer of learning: it sits on top of your usual training. In traditional machine learning algorithms, the set of parameters are learned for the specific task, whereas in Meta-Learning those parameters are optimized for any task presented. With training on a rich variety of base tasks, the model discovers general features of problems it can solve or types of features of a field which are promising, and modifies its own way of learning to fit new challenges [14].

There are two levels on which Meta-Learning takes place. At the task level the model’s inner parameters adapts to perform a specific task like image classification or object detection when assigned a specific environment. Usually, this fine-tuning is performed by an optimization algorithm such as gradient descent [15]. At the meta level, the system modifies the initial conditions of the model or the learning process itself so that it is flexible enough to quickly adapt to novel tasks. On this second level, processes such as finding the optimal hyperparameters, or creating architectures that promote adaptability, take place [16].

In the context of this work, Meta-Learning plays an essential role in enabling

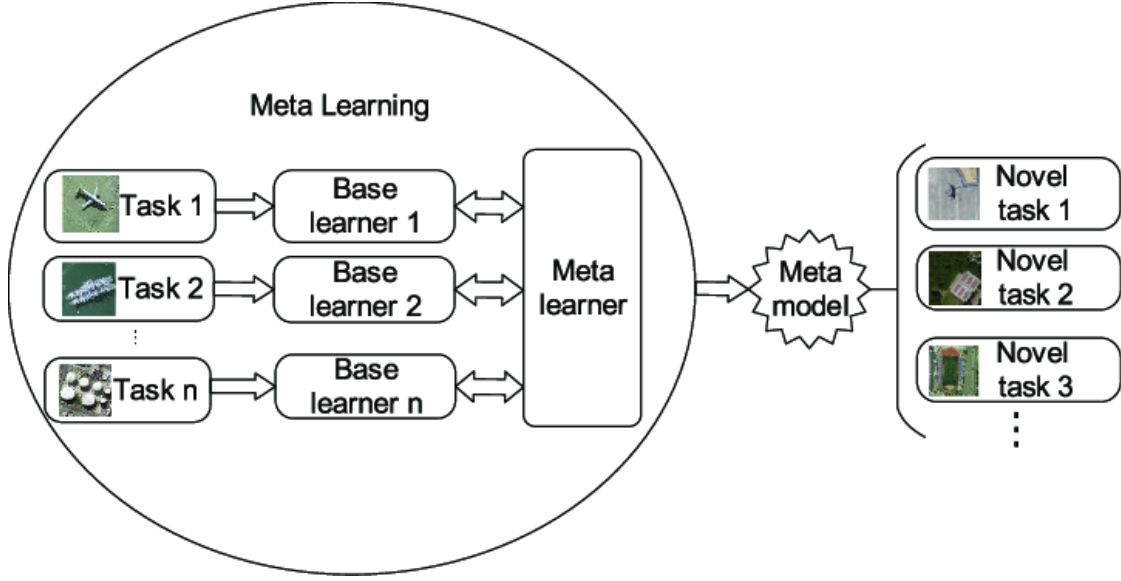


Figure 2.4: Diagram of Meta-Learning Architecture: A visual representation illustrating how a meta-learning model processes multiple tasks to learn general patterns that facilitate adaptation to new tasks. Such diagrams help in understanding the structure and internal functioning of meta-learning models.

robots, such as robotic arms, to adapt to new grasping and manipulation tasks in dynamic environments. In this way, robots can manipulate objects or situations not anticipated at the time of their initial training and learn quickly from a limited number of interactions. Furthermore, combining pre-trained visual representations, i.e., those trained using DINO, with Bayesian optimization methods enables the inclusion of accumulated contextual information. This not only makes manipulation more effective but also enables generalization to unseen situations, and hence, transitions between known and new tasks, to be smoother.

Meta-Learning has huge potential, but also significant limitations and challenges to overcome. This makes it one of the biggest challenges: scalability. Meta-Learning systems typically need massive computational resources to learn over many different tasks. When the complexity and/or number of tasks is increased, the computational cost becomes exponential, and effective application of Meta-Learning to large-scale real-world problems is less achievable.

Another significant challenge lies in the quality and diversity of the base tasks. For a Meta-Learning model to generalize effectively, the tasks used during training must cover a wide and representative range of scenarios. However, obtaining such a diverse and high-quality dataset can be expensive and time-consuming. Moreover, the model's performance can degrade if the tasks encountered during deployment differ significantly from those seen during training, leading to limited adaptability.

The balance between adaptation and generalization also poses a critical problem.

While the model aims to adapt quickly to new tasks, excessive focus on task-specific optimization can lead to overfitting [12], reducing its ability to generalize across different domains. Achieving an optimal trade-off requires careful design and fine-tuning of the Meta-Learning architecture.

And then there are debugging and interpretability concerns. Meta-Learning models are treated as black boxes, so it is hard to observe how and why they are going wrong in some cases. The lack of transparency makes it harder to fine-tune their performance and correct their faults.

Applications of Meta-Learning in robotics are numerous and promising. By accelerating adaptation to new tasks, computational efficiency is optimized, and robots are enabled to operate effectively in unstructured and variable environments. This is crucial in scenarios where available data is limited or where rapid adaptability is essential.

## 2.6 Bayesian Optimization

*Bayesian Optimization* (BO) is a probabilistic framework for optimizing an unknown, expensive-to-evaluate objective function  $f : X \rightarrow \mathbb{R}$ , where  $X \subseteq \mathbb{R}^d$  is a compact  $d$ -dimensional search space. The goal is to identify the global optimum  $x^*$ :

$$x^* = \arg \max_{x \in X} f(x),$$

using as few evaluations as possible [17].

This approach is highly useful in scenarios where evaluating  $f(x)$  requires a huge amount of resources, such as some simulation environments or experiments [18], when there is noise in the evaluations [19], or in another case, when the functional form of  $f(x)$  is unknown, such as in black-box optimization problems.

The procedure leverages prior knowledge and observations of the function  $f(x)$  to iteratively select new points for evaluation. It comprises two primary stages. First, in the learning stage, a *surrogate model* of  $f(x)$  is constructed or updated using previous evaluations [20]. In this work, the surrogate model is a Gaussian Process (GP), with a zero mean function  $\mu(x) = 0$  assumed for simplicity (see Section 2.6.1). Second, in the decision stage, an *acquisition function*  $\alpha(x)$  is employed to identify the next query point  $x_{t+1}$  [21]. Specifically, the next point is selected as

$$x_{t+1} = \arg \max_{x \in X} \alpha(x),$$

where  $\alpha(x)$  reflects the potential utility or promise of evaluating  $f$  at  $x$ .

Ultimately, the candidate solution is typically the point with the highest observed value of  $f(x)$  among all evaluated points [22].

## 2.6.1 Gaussian Processes

*Gaussian Processes* (GPs) are commonly used as surrogate models in Bayesian Optimization due to their ability to provide predictions and uncertainty estimates [20]. Formally, a GP is a collection of random variables where any finite subset follows a multivariate Gaussian distribution.

A GP is fully characterized by:

$$f(x) \sim \mathcal{GP}(\mu(x), k(x, x')),$$

where  $\mu(x)$  is the mean function and  $k(x, x')$  is the covariance (kernel) function.

### Mean function and kernel function

The mean function  $\mu(x)$  represents the expected value of  $f(x)$ . In many applications,  $\mu(x) = 0$  is assumed to simplify the mathematical treatment [20]. In our project, we have explicitly adopted the assumption that  $\mu(x) = 0$  for all  $x$ .

The kernel function  $k(x, x')$  captures the covariance between  $f(x)$  and  $f(x')$ . This function allows us to model the smoothness and structure of the objective function. Some commonly used kernels include the RBF (or Squared Exponential) kernel and the Matérn kernel. The RBF kernel is defined as

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

where  $\ell$  is the length scale and  $\sigma^2$  is the process variance. The Matérn kernel of order 5/2 is given by

$$k_{\text{Matérn}}(x, x') = \sigma^2 \left(1 + \frac{\sqrt{5}r}{\ell} + \frac{5r^2}{3\ell^2}\right) \exp\left(-\frac{\sqrt{5}r}{\ell}\right),$$

where  $r = \|x - x'\|$ .

### Posterior distribution

Suppose we have made  $t$  observations:  $D_t = \{(x_i, y_i)\}_{i=1}^t$ , where  $y_i = f(x_i) + \varepsilon$  and  $\varepsilon \sim \mathcal{N}(0, \sigma_n^2)$  is Gaussian noise with variance  $\sigma_n^2$ . Assuming a zero mean function, the *posterior* distribution of the GP at a new point  $x^*$  is also Gaussian:

$$f(x^*) \mid D_t \sim \mathcal{N}(\mu_t(x^*), \sigma_t^2(x^*)),$$

where:

$$\mu_t(x^*) = k_*^\top (K + \sigma_n^2 I)^{-1} y, \tag{2.1}$$

$$\sigma_t^2(x^*) = k(x^*, x^*) - k_*^\top (K + \sigma_n^2 I)^{-1} k_*, \tag{2.2}$$

and:



- $k_* = [k(x^*, x_1), k(x^*, x_2), \dots, k(x^*, x_t)]^\top$  is the vector of covariances between the new point  $x^*$  and the already observed points.
- $K$  is the covariance (or *kernel*) matrix of size  $t \times t$ , with elements  $K_{ij} = k(x_i, x_j)$ .
- $\sigma_n^2$  is the noise variance. To incorporate noise into the model,  $\sigma_n^2 I$  is added to the matrix  $K$ .
- $I$  is the  $t \times t$  identity matrix.
- $y = [y_1, y_2, \dots, y_t]^\top$  are the noisy observations.

This result is obtained by *conditioning* the initial Gaussian distribution ( $f(x) \sim \mathcal{GP}(\mu, k)$ ) on the collected observations. The fact that Gaussian Processes retain their distributional form (they remain Gaussian) after observing data is one of their main advantages.

## 2.6.2 Acquisition functions

To determine the next evaluation point  $x_{t+1}$ , an *acquisition function*  $\alpha(x)$  is defined. The new query point is typically chosen as:

$$x_{t+1} = \arg \max_{x \in X} \alpha(x).$$

Hence, the acquisition function directly drives the **Decision stage**: once the GP (surrogate model) is updated (the **Learning stage**), we compute  $\alpha(x)$  and select the point  $x_{t+1} = \arg \max \alpha(x)$ . Some of the most widely used acquisition functions are:

- **Probability of Improvement (PI)**:

$$\alpha_{\text{PI}}(x) = \Phi\left(\frac{\mu_t(x) - f_{\text{best}}}{\sigma_t(x)}\right),$$

where  $f_{\text{best}}$  is the best value observed so far,  $\mu_t(x)$  and  $\sigma_t(x)$  are the *posterior* mean and standard deviation at  $x$ , and  $\Phi$  is the cumulative distribution function (CDF) of the standard normal distribution [23].

- **Expected Improvement (EI)**:

$$\alpha_{\text{EI}}(x) = (\mu_t(x) - f_{\text{best}}) \Phi(Z) + \sigma_t(x) \phi(Z),$$

where

$$Z = \frac{\mu_t(x) - f_{\text{best}}}{\sigma_t(x)}, \quad \phi(Z) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{Z^2}{2}\right), \quad \Phi(Z) = \int_{-\infty}^Z \phi(u) du.$$

- $\mu_t(x)$  is the *posterior* mean at  $x$ .
- $\sigma_t(x)$  is the *posterior* standard deviation at  $x$ .
- $f_{\text{best}}$  is the best observed value of the function so far.
- $\phi(\cdot)$  and  $\Phi(\cdot)$  are, respectively, the *probability density function* (PDF) and the *cumulative distribution function* (CDF) of the standard normal distribution.

EI combines two terms:

$$(\mu_t(x) - f_{\text{best}}) \Phi(Z) \quad \text{and} \quad \sigma_t(x) \phi(Z).$$

The first measures the expected improvement over  $f_{\text{best}}$  (if the mean  $\mu_t(x)$  is higher), and the second captures the contribution from uncertainty. This property allows exploring less-known areas (when  $\sigma_t(x)$  is large) and refining near the most promising regions (when  $\mu_t(x)$  is high) in a balanced manner [21].

- **Upper Confidence Bound (UCB):**

$$\alpha_{\text{UCB}}(x) = \mu_t(x) + \beta \sigma_t(x),$$

where the parameter  $\beta$  controls the balance between exploration and exploitation [24].

## Acquisition function selection and Algorithmic Outline

For the present thesis, the  $\alpha_{EI}(x)$  (Expected Improvement) acquisition function is employed. This choice is motivated by several factors. First, it provides a clear balance between exploring uncertain areas, where  $\sigma_t(x)$  is large, and exploiting areas where the mean  $\mu_t(x)$  is high. Additionally, it has demonstrated good performance across a wide variety of global optimization problems. Finally, it is relatively straightforward to calculate and interpret, as it combines the potential improvement with the probability that such improvement will occur.

After computing  $\alpha_{EI}(x)$  at the points of interest (or, alternatively, using global optimization methods on the acquisition), the next sampling point is selected:

$$x_{t+1} = \arg \max_{x \in X} \alpha_{EI}(x).$$

Next, the Gaussian Process (GP) model is fit or updated using the data  $\mathcal{D}_t$  in the learning stage. The objective function  $f(x_{t+1})$  is then evaluated, the observations are updated as  $\mathcal{D}_{t+1}$ , and the process is repeated in alternating learning and decision stages until the stopping criteria are met (for example, a maximum number of evaluations or convergence of the function value).

### 2.6.3 Incorporating Context Variables in Bayesian Optimization

In Bayesian Optimization (BO), **context variables** are parameters that are not optimized directly. Instead, they capture additional information about the surrounding environment, conditions, or constraints where the objective function  $f(x, c)$  is evaluated. In this setup,  $x \in \mathbb{R}^d$  represents the decision variables we aim to optimize, while  $c \in \mathbb{R}^k$  denotes the fixed context variables. Including these contextual factors in the model often leads to improved predictive accuracy because it helps account for external influences that affect how the function behaves.

Context variables are especially helpful in scenarios when the system simulates under different conditions. As an example, one such focus area can be in hyperparameter tuning for Machine Learning models, where multiple hardware configurations or multiple environment settings such as GPU type or GPU memory can emerge as context variables or dimensions [25]. In dynamic systems, also, temperature, humidity, or time of day may be relevant. In personalized scenarios, context can also be user-specific (demographics, preferences, etc.).

To incorporate context variables, the surrogate model, typically a Gaussian Process (GP), is trained using inputs  $(x, c)$  and outputs  $f(x, c)$ . During optimization, the context variables  $c$  are held fixed, and the acquisition function  $\alpha(x; c)$  guides the selection of the next query point:

$$x_{t+1} = \arg \max_{x \in X} \alpha(x; c),$$

where  $\alpha(x; c)$  explicitly includes the context information. As evaluations are performed, the dataset  $D_t = \{(x_i, c, f(x_i, c))\}_{i=1}^t$  is updated to improve the model's understanding of the relationship between  $x$ ,  $c$ , and  $f$ .

The implementation of context variables has major advantages. Not only do they improve the surrogate model's prediction of the objective function and help improve interpretability through consideration of external influences, but they also allow BO to perform in real-world situations where external or systemic factors play an important role. It is because adding context variables will also increase the dimensionality of input space and that can bring computational issues. Thus, context variables must be carefully designed to be useful and relevant.

For example, the hyperparameters of a neural network could be optimized for datasets with varying characteristics (e.g., varying numbers of samples or features). you can consider these dataset characteristics as context variables and use them to enable the BO framework to better predict the model performance given dataset conditions.

This has the advantage that BO can not only find the best hyperparameters but also takes into account the even specific context how it is used the model.

## 2.7 Advanced Modeling: Deep Kernels and the ADFK-IFT Framework

### 2.7.1 Fundamentals of Deep Kernels

Deep Kernels enables us to combine the powers of deep learning with the probabilistic nature of Gaussian Processes. By using transformations generated by neural networks themselves, they enable the model to learn lots of rich and nonlinear representations of the input data [26]. The derived kernel models complex relationships in high-dimensional, heterogeneous spaces, enabling the model to capture features across multiple sources of data; by treating visual, spatial, and contextual inputs as an extension of the same kernel domain. The combination of such deep feature extraction with Gaussian Process regression allows for a systematic trade-off between expressiveness and uncertainty estimation, improving performance in applications demanding strong probabilistic predictions.

### 2.7.2 Approximate Deep Feature Kernels (ADFK)

The **ADFK** approach enables efficient processing of deep features in Gaussian Processes. Rather than exhaustively computing pairwise similarities along all possible pairs of features, a neural network is employed to capture the input in a condensed embedding. That simplifies calculations while preserving the most important properties in the data.

Formally, the approximate kernel is expressed as:

$$k(x, x') = \psi(x)^T \psi(x'),$$

where:

- $\psi(x) = \text{MLP}(\phi(x))$ : A deep transformation of features  $\phi(x)$  that produces a compact representation [26].
- The kernel computation utilizes this reduced subspace to approximate relationships between deep features more efficiently.

By integrating multiple data modalities within a single optimization model, ADFK effectively addresses the computational challenges posed by high-dimensional inputs. In doing so, it preserves critical relationships and accelerates tasks such as regression and classification in complex domains.

### 2.7.3 Independent Feature Transformations (IFT)

Independent Feature Transformations are a method to deal with heterogeneous input types — including images, spatial coordinates, or text data — in a unified way. The process of transforming the input depends on the kind of input, or modality, such as images or text, and generally, specialized networks or filters are used for each type of input. These transformed representations are subsequently combined into a unified feature vector for subsequent modeling [26]. In this way, we maintain the high importance of each data source while still incorporating the potential for new features. “By addressing modalities independently, the system gains better awareness of how they come together to achieve the desired outcome, which in turn leads to better generalization and robustness across different tasks.”

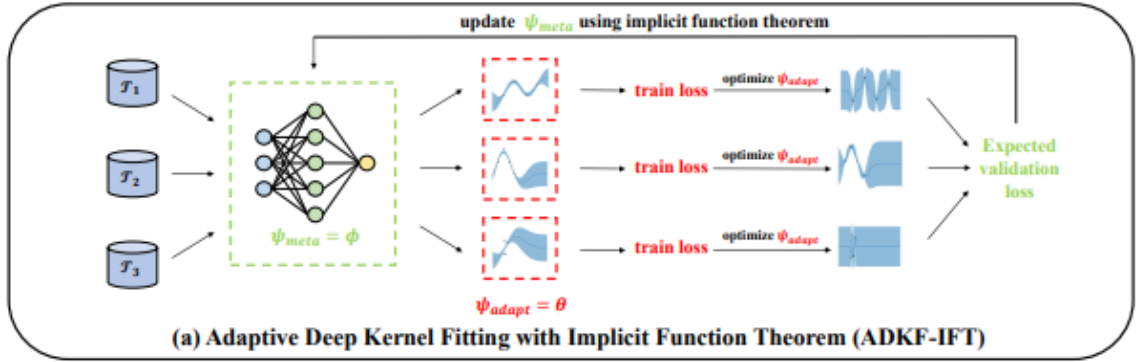


Figure 2.5: Schematic representation of Adaptive Deep Kernel Fitting with Implicit Function Theorem (ADKF-IFT), illustrating the meta-learning process [26].

**Applications to the Proposed Model.** The combination of Deep Kernels, ADFK, and IFT underpins several advantages for advanced optimization tasks. First, the reduction of computational costs through approximate deep features enables practical scalability, particularly when dealing with large, heterogeneous datasets. Second, the separation of feature transformations for different modalities guarantees that specialized methods handle each data source without interference. Finally, by extending Gaussian Processes with expressive neural representations, the model can capture nonlinear relationships and subtle dependencies, improving performance in tasks like predicting optimal configurations or identifying key interest points. These capabilities are highly relevant in robotic manipulation, where accurate modeling of uncertainties, faster convergence, and resilience to noise can lead to more reliable and efficient actions.



## Chapter 3

# Project Pipeline

The pipeline of the project lays out the sequential steps required to achieve the established objectives, serving as a structured roadmap for the integration and execution of its various components. It ensures a seamless flow of information and functionality, starting from the initial stages of data input and feature extraction, through the integration of visual and spatial metrics, to the final evaluation and optimization phases. Each step is carefully designed to align with the overarching goal of building a robust and context-aware robotic grasping system.

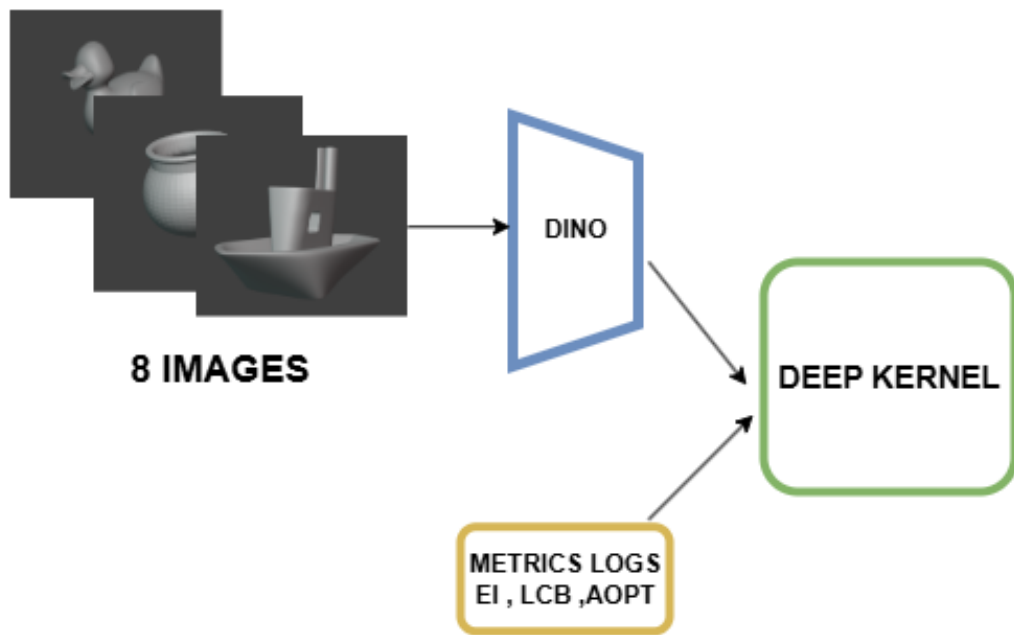


Figure 3.1: Training pipeline: Visual embeddings are extracted using DINO from 8 images of the objects, while metrics logs from Bayesian optimization (EI, LCB, cAopt) are integrated to generate a deep kernel.

### 3.1 Training Phase

While the training phase forms the backbone of this pipeline, the focus is primarily on how visual embeddings and spatial metrics are integrated to construct a robust grasping system. As illustrated in Figure 3.1, the process begins with the collection of input data, consisting of eight images of various objects, which serve as the foundation for feature extraction. These images are processed using the DINO framework to extract visual embeddings that capture essential object features and align visual and spatial data. Simultaneously, grasping metrics logs are generated using Bayesian optimization techniques, including acquisition functions such as **Expected Improvement (EI)**, **Lower Confidence Bound (LCB)**, and **Certainty-Aware Optimization (cAopt)**. These logs store valuable spatial and contextual information related to grasp success. The extracted visual embeddings and grasping metrics are then integrated into a unified deep kernel representation, which serves as the basis for subsequent optimization and evaluation stages. This integration is critical to ensure that visual and spatial features are properly aligned, enabling the system to generalize effectively across a diverse set of objects. This aligned representation is subsequently used in the optimization steps to achieve robust and high-performance grasping capabilities.

### 3.2 Context-Based Optimization Workflow

Context-based optimization workflow is an integral part which helps to maximize the grasp configurations for robot manipulation system. Through the integration of visual and contextual data, this process generates dynamic and efficient grasping policies that are well-suited for the specific properties of each object. It continually refines its ability to navigate and control diverse and dynamic environments through contextual modeling, machine learning methods, and Bayesian optimization.

The process starts by identifying the object’s context, using two main sources of information. First, the **current context of the object** considers the visual and physical properties of the object in its specific environment. Second, historical data from past optimization tasks and related queries is utilized. This combination not only offers a strong foundation but also avoids redundancy by reusing relevant information, making the initial stage more efficient. Based on these sources, an initial grasp metric is computed, referred to as the mean grasp metric (mean  $x.tr$ ), which is calculated from images of the object taken from eight different angles. This initial metric offers an overview of the average behavior of grasping strategies and establishes a foundation



for further optimization.

The optimization process uses a Bayesian approach that combines probabilistic modeling and deep learning techniques to improve grasp configurations step by step.

In the first phase, a model called a Gaussian Process (GP) is either trained or updated. This model uses a deep kernel, which incorporates pretrained weights from a Multi-Layer Perceptron (MLP) [27]. The kernel is crucial because it captures the visual and spatial properties of the object, creating a strong contextual representation that aligns the key features of the object. This representation is vital for accurately understanding the unique characteristics of the object while making the optimization process more efficient [28].

After the context is established, the next phase involves performing the Bayesian optimization itself. During this stage, an acquisition function, typically the Expected Improvement (EI), is computed. This function uses the contextual representation to pinpoint promising grasp configurations, balancing two goals: exploring new configurations and refining existing ones. Guided by the updated GP model, the acquisition function identifies the next query point that maximizes  $\alpha(x)$ , ensuring the system moves closer to finding the best possible configurations [29].

The selected configuration is evaluated against the objective function, which measures the success of the grasp. The observed results  $(x, f(x))$  are incorporated into the existing dataset, thereby updating the GP model to reflect the most recent information. This approach is highly iterative, which allows the learning of grasp strategies to be continuously improved. At every iteration, instead of taking the images as fixed, the model continuously updates its understanding of image configurations by adding new data, enabling it to learn the unique characteristics of each object. Moreover, the possibility to load queries and contextual data from previously evaluated objects, greatly reduces the time overhead by eliminating redundancy and makes the system adaptable to similar problems [29].

It is an interesting regime also because it is pretty good at using existing knowledge and also allows for dynamic updating. With the usage of superior visual processing, contextual modeling and iterative optimization techniques the system is not only refined on an ongoing basis but also appears to be substantial in future implementation for robotic manipulation tasks irrespective of its environment conditions. The ability to manage complex objects, changing environmental conditions, and learning new tasks rapidly. The approach is efficient and highly adaptive due to its reuse of previous knowledge and use of deep kernels based on neural networks.

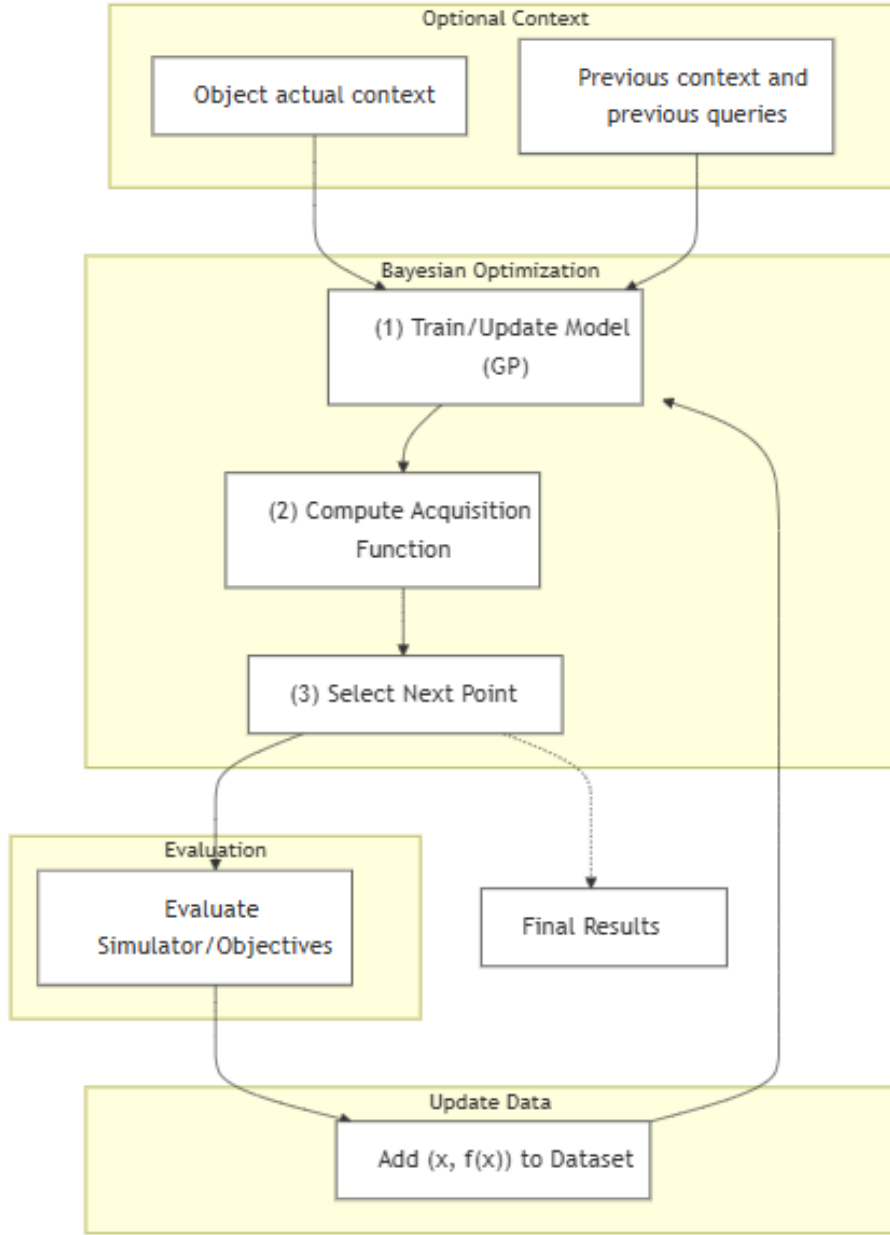


Figure 3.2: Context-Based Object Optimization Pipeline scheme

Finally, this pipeline exemplifies an advanced approach to robotic manipulation, bringing together vision, context-based modeling, and Bayesian optimization into a unified, small-footprint system. The nature of iterative refinement, together with the reuse of knowledge as well as stability to each object’s specific characteristics, consistently allows for performance in dynamic and demanding environments. With the potential to adapt over time, this novel take opens new domains of robotic automation and raises the bar on the ability to marry advanced technology to the practical world.

# Chapter 4

## Setup

### 4.1 Description of Objects

The dataset used in this project consists of 8 representative solid objects with simple geometric shapes, specifically designed to evaluate robotic grasping strategies. Each object has specific features that influence the difficulty of manipulation, providing a range of complexity to help generalize the developed strategies.

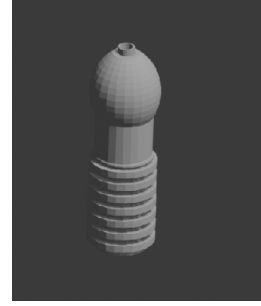
Several additional candidate objects were tested before this dataset was determined as final. Either their inclusion had an adverse effect on the results or they were contradictory to study aims. Specifically, the preselection process was guided by the aim of this task of coming up with objects that were both feasible to grasp and meaningful challenges to consider, so that it would end up being a challenging, and yet realistic and doable, dataset to test the system on, given the numerous potential manipulation tasks to be carried out.

The final selection of objects includes the following:

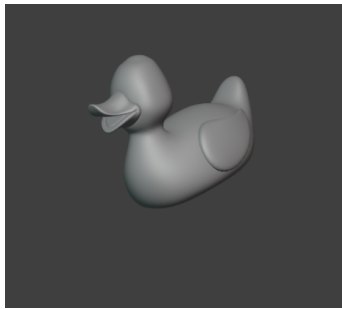
- **Jar:** A cylindrical object with a wide base, making it relatively easy to grasp due to its stability and uniform structure.
- **Bottle:** Another cylindrical object, but taller and narrower, increasing the difficulty due to its higher center of gravity.
- **Duck:** An irregular shape with curved edges, challenging the system’s ability to handle complex geometries.
- **Boat:** A structure with a flat base and narrower sections, testing stability during grasping.
- **Trophy:** Tall and detailed at the top, requiring precise grasping configurations.
- **Mouse:** Compact and curved, with a slippery surface that makes achieving a robust grasp more difficult.



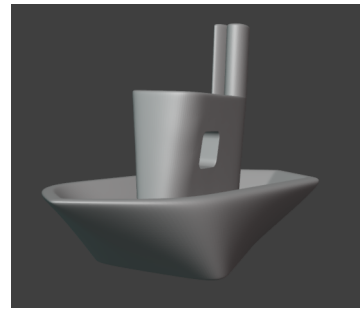
Visual Image for Dino of Jar



Visual Image for Dino of Bottle



Visual Image for Dino of Duck



Visual Image for Dino of Boat



Visual Image for Dino of Trophy



Visual Image for Dino of Mouse



Visual Image for Dino of Charger



Visual Image for Dino of Ice Cream

Figure 4.1: Visual representations of the objects (Jar, Bottle, Duck, Boat, Trophy, Mouse, Charger, and Ice Cream) processed with Dino.

- **Charger:** Elongated with angular shapes, representative of everyday objects.
- **Ice Cream:** A cone-shaped structure with a narrow base, designed to test unstable grasping conditions.

The use of Blender to render these objects ensures visual quality consistency and real geometry. And a consistent rendering for the camera settings, to keep true proportions. The same fixed camera position and orientation was used to image all objects so that every object was visualised at the same angle, and under the same lighting. Such strategies significantly restrain the variability of visual inputs induced by differing viewing angles and consequently, make the extracted visual characteristics invariant and comparable throughout the dataset.

This results in a dataset with a high degree of uniformity by maintaining these standardized rendering conditions, which is essential for aligning the visual features with spatial grasping metrics. By producing coherent representations, perspective continuity also eases the downstream feature extraction process by allowing the model to concentrate on object-level properties instead of being influenced by the visual inconsistencies that contribute nothing to learning.

#### 4.1.1 Simulation Environment: Simox

The simulation framework that was utilized in this project to create and evaluate the robotic grasping task is called simox. Such simulation models capture the complete physical representation of the robot and objects of interest in 3D space and help to analyze and define the quality of grasp configurations in controlled experimental conditions. Specifically, Simox is utilized in this work to simulate grasping scenarios with the humanoid robot iCub and evaluate the grasp quality of the selected objects according to pre-defined metrics. Such a setup makes realistic yet reproducible experiments possible, while offering straightforward integration of visual and spatial data for optimization [30].

#### 4.1.2 iCub Robot

The iCub humanoid robot, central to this project, is a research platform specifically designed for advancing studies in embodied AI. Standing at 104 cm and weighing 30 kg, it is comparable in size to a five-year-old child. The robot features a highly articulated structure with 53 degrees of freedom (DoF), enabling it to perform complex manipulation tasks [31]. Its hands are equipped with 9 DoF, supporting precise and dexterous object interactions, while over 3,000 tactile sensors distributed across its

body enable robust perception of the environment. Additionally, the iCub is equipped with cameras, microphones, and force/torque sensors, which enhance its ability to interact dynamically with objects. Table 4.1 summarizes its degrees of freedom, highlighting its suitability for the manipulation and grasping tasks explored in this project.

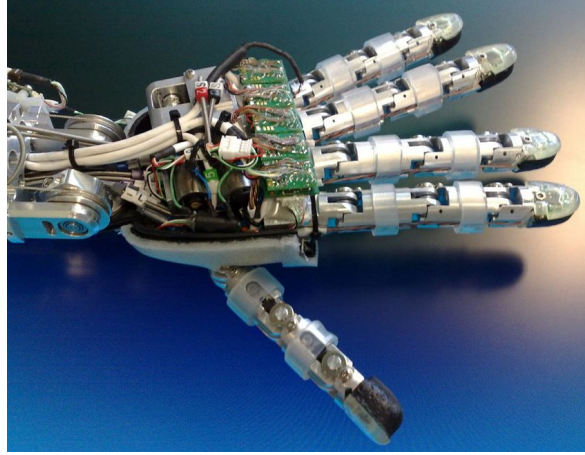


Figure 4.2: The iCub Robot.

Component	DoF	Description
Eyes	3	Independent vergence and tilt movements.
Head	3	Full neck mobility, including tilt, swing, and pan.
Chest	3	Tilt, swing, and pan movements.
Arms	7 (each)	Distributed across shoulder (3 DoF), elbow (1 DoF), and wrist (3 DoF).
Hands	9	Includes 19 joints with various coupled and independent configurations.
Legs	6 (each)	Sufficient for walking motions.

Tabla 4.1: Degrees of Freedom (DoF) of the iCub Robot.

### 4.1.3 BayesOpt

BayesOpt library is the selected among a bunch of libraries , to be utilized in this project to implement Bayesian Optimization to optimize grasping strategies [32]. BayesOpt is an efficient framework specifically designed for Bayesian Optimization of black-box functions, i.e., the grasp quality function in this project, where the evaluations are computationally expensive and gradients are not directly available.

In the context of this project, BayesOpt serves to optimize grasp configurations iteratively by leveraging a Gaussian Process (GP) as the surrogate model. The library is integrated into the pipeline to guide the optimization process by predicting the

performance of potential grasp configurations and suggesting improvements based on both the expected performance and uncertainty. This approach is particularly advantageous for robotic tasks where each evaluation of grasp quality involves simulations or physical testing.

The procedure begins by initializing the GP model with an initial batch of random samples to establish the baseline for each object. The samples are selected to cover the space of grasp configurations uniformly, giving a diverse starting point for optimization. We use the acquisition function, in this case Expected Improvement (EI), to pick and evaluate new grasp configurations at each iteration. BayesOpt provides flexible and efficient mechanisms to manage these computations and is therefore well adapted to the iterative nature of this research.

One of the main advantages of BayesOpt is that it is able to handle the high-dimensional input formed by the concatenation of vision embeddings and spatial coordinates. Wrapping BayesOpt in the pipeline also allows it to interface naturally with the Gaussian Process model that we have in JAX. The library supports extensive logging during optimization of configurations that were tried, e.g., spatial coordinates of the grasps, their success probabilities, and acquisition function outputs.

The use of BayesOpt in this work enables the exploration and exploitation of the grasp configuration space efficiently so that the optimization converges to effective and robust strategies for every object. It is very computationally efficient and also flexible, allowing iterative refinement, which aligns the grasp quality metrics and the visual and spatial features discovered from the dataset.

#### 4.1.4 JAX for Deep Kernel Learning

JAX, a high-performance numerical computing library, is employed in this project to implement and optimize the Gaussian Process used for modeling grasp quality. The input features for the model combine visual embeddings, extracted from DINO, with spatial information representing the grasp positions in three-dimensional space. Specifically, each grasp is represented by a 384-dimensional visual feature vector and a 3D spatial vector, which are merged into a single feature vector for processing.

JAX offers significant advantages for this application. Its automatic differentiation capabilities allow for efficient optimization of the Gaussian Process parameters, ensuring that the model adapts to the data effectively. Additionally, JAX’s ability to perform computations on GPUs accelerates the training process, enabling scalable and fast evaluation of the Gaussian Process even with high-dimensional input data. This computational efficiency is essential for managing the complexity of the optimization process and ensuring timely convergence.

The integration of JAX within the workflow also facilitates seamless interaction with other components of the pipeline, such as Bayesian Optimization. By leveraging JAX’s computational capabilities, the Gaussian Process is able to accurately model the relationships between grasp configurations and their success probabilities. This alignment between spatial and visual data is crucial for developing grasping strategies that generalize effectively across diverse objects and scenarios.



# Chapter 5

## Metodology

### 5.1 Data Generation and Preprocessing

The first step in the training process involves generating training data from visual features and spatial grasping metrics.

#### 5.1.1 Extraction of Visual Features with DINO

The DINO model (Distilled Knowledge from Self-Supervised Learning) was used to extract the visual features used in this project. DINO input images were created by rendering the selected objects within Blender, as described in the Setup section. It ensured that the visual data across all the objects in terms of quality orientation and perspective was constant. Such rendering standards allowed the comparison of the visual features extracted to the spatial grasping measurements acquired from Simox.

The camera setup in Blender was mentioned specifically as it gave a fair and a normalized vision of each object. Such transformations as well as maintaining the camera's position, orientation and camera parameters throughout the rendering process was done to ensure that there will be proportionality and non-influence variation in visual inputs which may have affected the feature extraction process. This uniformity was vital for the DINO model to devote its attention to the intrinsic appearance and geometric properties of the objects as opposed to irrelevant ones, such as perspective distortions or alterations to lighting.

From these standardized images, DINO generated 384-dimensional embedding vectors that encapsulate the visual features of each object in a high-level and compact form. These embeddings form a vital part of the dataset as they ensure the visual information is rich and well-structured prior to its fusion with the spatial measures.

This process of combining Blender-rendered visual data with Simox-generated spatial metrics establishes a dependable foundation for the training pipeline. By aligning these two datasets, the project ensures that the learning model operates on

coherent and complementary inputs, ultimately supporting the development of effective robotic grasping strategies.

### 5.1.2 Generation of Grasping Metrics and Logs

Grasp metrics were obtained using Bayesian optimization in Bayesopt to complete the learned visual features of DINO. These metrics provide a rich context, as they summarize the spatial and contextual characteristics of successful grasps, forming a solid base for visually and spatially matching datasets. The optimization used a variety of acquisition functions with varying exploration versus exploitation of the grasping space. The details of the process are as follow:

Bayesian optimization was employed for every object to update iteratively the grasp positions and their respective success probabilities. The target of the optimization was to maximize grasp quality, which was measured as a proxy by a volume-based metric using the grasp success rate. The procedure began with an initialization step, which served as a baseline for all optimization runs by randomly sampling an initial set of grasps. After initialization, the optimization procedure updated the grasping parameters at each iteration, progressively enhancing the configurations based on feedback from the surrogate model.

Optimization relied on various acquisition functions for effective exploration and exploitation of the grasping space. Expected Improvement (EI) was employed for 100 iterations with 30 initialization samples and 70 optimization steps. Similarly, the Lower Confidence Bound (LCB) acquisition function was used with the same number of iterations, including 30 samples for initialization and 70 steps for optimization. Certainty-Aware Optimization (cAopt) used fewer iterations, with 2 samples for initialization and 30 steps for optimization, where it focused on specific scenarios that required precise adjustments.

During the optimization process, logs were generated at each step to document the grasping metrics. These logs included spatial coordinates  $[x, y, z]$ , grasp success rates, and contextual information about the chosen acquisition function. Each entry was stored in JSON format, ensuring a consistent structure and straightforward integration with the extracted visual features. The Bayesian optimization results highlighted the top-performing grasping configurations for each object, preserving a reliable alignment with their respective visual embeddings.

The grasping metrics were further analyzed to classify grasps into distinct categories. Good grasps were defined as those with a quality score greater than 0.3, demonstrating sufficient alignment between spatial and visual features, which resulted in high grasp success probabilities. Conversely, poor grasps, identified by a quality

score below 0.3, represented configurations that often failed to achieve robust spatial contact, though some specific exceptions were observed.

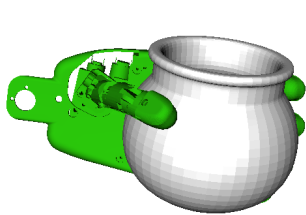
Grasps with a quality score higher than 0.3 are considered as good grasps. These configurations offer good grasp success probabilities due to sufficient alignment of the visual and spatial features. On the other hand, poor grasps are those with a quality score less than 0.3. In general, these configurations do not reach stable spatial contact, although individual exceptions may take place in certain cases.

This optimization aims to find one or more close to optimal grasping configurations which can produce a quality score most of the time above 0.4. Logs and metrics from this stage allow us to track how these configurations change over the course of the optimization process, providing confidence that acquisition and execution of grasping strategies are performed in a manner that is reliable and consistent with the visual embeddings extracted via DINO.

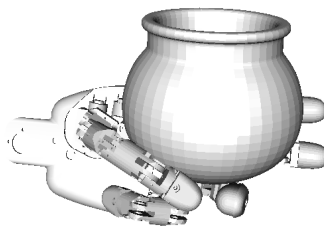
To measure how well the Expected Improvement (EI) acquisition function works, the grasp quality is plotted over multiple iterations for each object. These plots demonstrate how grasp quality changes during optimization, emphasizing EI's role in fine-tuning grasping configurations. By examining the trends in these graphs, it becomes evident whether the optimization process converges and meets the desired quality target of (0.3–0.4) for each object. This visual data also provides a clear picture of the effectiveness and robustness of the Bayesian optimization strategy.

### 5.1.3 Examples of Good and Poor Grasps

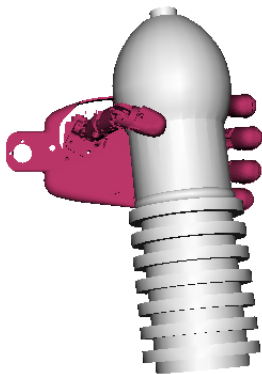
To illustrate the results of the Bayesian optimization process, examples of both good grasps (quality  $> 0.3$ ) and poor grasps (quality  $< 0.3$ ) are provided for each object. These examples demonstrate grasps with different spatial arrangements resulting in successful and unsuccessful grasps.



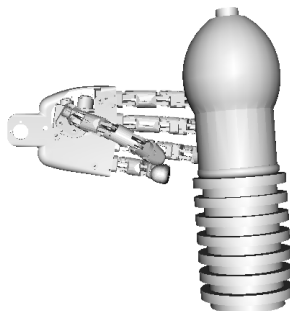
(a) Good Grasp: Jar



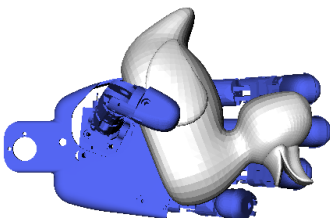
(b) Poor Grasp: Jar



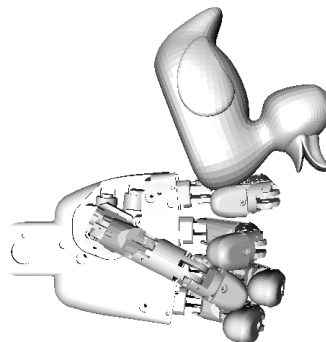
(c) Good Grasp: Bottle



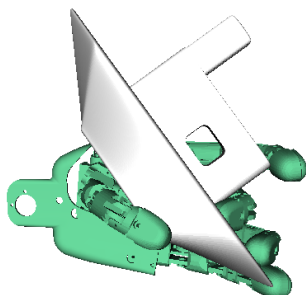
(d) Poor Grasp: Bottle



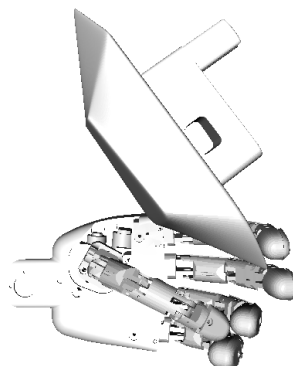
(e) Good Grasp: Duck



(f) Poor Grasp: Duck



(g) Good Grasp: Boat



(h) Poor Grasp: Boat

Figure 5.1: Examples of good and poor grasps for different objects. Good grasps (quality  $> 0.3$ ) are on the left, while poor grasps (quality  $< 0.3$ ) are on the right. The objective is to achieve high-quality grasps (target  $> 0.4$ ) across all objects.

### 5.1.4 Loading and Combining Data

After generating the visual features, they were combined with the grasping metrics to create a unified dataset. The grasping metrics for each object were represented by spatial coordinates  $([x, y, z])$  and associated success rates, with all data stored in JSON files for consistency and ease of access. The creation of the final dataset involved several steps. First, the JSON data for each object was loaded, containing grasping metrics such as positions and success probabilities. Then, the DINO features corresponding to the rendered images of the objects were retrieved. Finally, these two sources were combined into a single dataset consisting of input-output pairs  $(X, y)$ , where the inputs represented the combined visual and spatial features, and the outputs reflected the associated grasp success rates.

### 5.1.5 Data Splitting

Once the dataset was prepared, it was split into three distinct subsets to enable good model training and evaluation. The training set, accounting for 80% of the data, was used to tune the kernel and feature extractor. The validation set, accounting for 10% of the data, was used to tune hyperparameters and evaluate intermediate performance throughout training. After that, the test set, which was also 10% of the data, was kept aside for assessing the model’s generalization to new objects and tasks. This approach offers a comprehensive assessment of the model while maintaining the consistency of the data across the visual and spatial domains throughout.

## 5.2 Deep Kernel Learning with JAX

### 5.2.1 Input to JAX

For each object, the training data ( $X_{\text{train}}$  and  $y_{\text{train}}$ ) is passed to the model for training. The features consist of a combination of:

- **Visual Features:** 384-dimensional embeddings extracted from DINO.
- **Spatial Features:** 3D coordinates representing the positions of the grasps in space:  $x, y, z$ . All the features are then combined in one input vector for every grasp. This gives, for every sample, a feature vector of size 387. This is used by JAX for training the Gaussian Process with a learned kernel, adapting to the object characteristics.

By using this approach, it is ensured a thorough evaluation process that keeps the visual and spatial data aligned in a consistent way.

## 5.2.2 Metrics and Kernel in JAX

The Gaussian Process model uses a custom kernel based on the Matern 5/2 covariance function, implemented with the JAX library for high-performance computation. This kernel incorporates a length scale parameter, which controls the relative weighting of different feature dimensions by scaling the input features. A model amplitude parameter that governs the general magnitude of covariance-allows capturing variation in signal strength between datasets. The kernel gives similarities between feature vectors through the L2 distance metric, providing a robust smooth measure of their mutual correlation. The amplitude and length scale are parameters learned from training in order to optimize the performance of this kernel for best modeling of the complex relationship between the visual and spatial grasping features. This choice of kernel can allow great flexibility and precision for capturing variability and patterns across diverse objects.

## 5.2.3 Execution of Gaussian Process

In this work, Gaussian Processes (GP) were used for tuning the kernel parameters of the deep kernel. 232 evaluations were performed using the acquisition functions described above for optimizing the kernel hyperparameters. These 232 evaluations were for optimizing the predictive performance of the model for robotic grasping by the proper leveraging of visual and spatial features. The hyperparameters that were tuned during the execution of deep kernel GP were the amplitude, length scale, and noise parameters of the deep kernel GP, as well as the meta-parameters of the MLP feature extractor, which were tuned for improved feature representation and numerical stability of integration of the input data.

## 5.2.4 Incorporating Contextual Features

### Feature Extraction for Context

After training the deep kernel GP, It is computed contextual features for each object based on the learned grasping characteristics. For a given object, It is calculated the mean of all grasp success metrics associated with the object. This mean, referred to as the **grasp metric mean**, acts as a representative summary of the object’s grasping properties. The process is as follows:

1. For each object, all grasp success values (e.g., grasp volume metrics) are aggregated to compute their mean.

2. The average grasp metric is passed through the MLP feature extractor, resulting in a **context vector** that encodes high-level information about the object.
3. This context vector is then added as an extra input to Bayesian optimization, enabling context-aware optimization.

The average grasp metric used ensures that the context vector summarizes, in general, how hard or easy it is to grab each object. This is important information that guides the process of optimization.

### Bayesian Optimization with Context

Then the deep kernel GP is trained, and it can derive contextual features using the learned grasping properties for each object accordingly. It averages all the measures of grasp success with respect to the object for the target object. This mean is called the grasp metric mean, and it is a representative summary of the grasping features of the object. The process is as follows:

1. At each iteration, the context vector for the current object is combined with the standard kernel parameters.
2. The acquisition functions (EI, cAopt, and LCB) employ the context-enhanced parameters to guide the search for optimal solutions.
3. The optimization process adapts dynamically to the distinct characteristics of each object, improving its ability to generalize across a range of grasping scenarios.

This method ensures that the optimization process remains robust and is precisely tailored to the unique grasping properties of every object.

## 5.3 Training Phases and Context Extraction

### 5.3.1 Adaptation Phase

The adaptation phase focuses on tuning the Gaussian Process (GP) parameters for each object. This process involves several key steps. First, the object-specific data, including the training, validation, and test sets, is loaded. Then, features for each object are extracted using the MLP feature extractor, ensuring that the relevant visual and spatial characteristics are adequately represented. Finally, the GP parameters, such as amplitude, noise, and length scale, are updated based on the adaptation loss. At this point, unique grasping patterns are identified for each object, allowing the GP kernel to efficiently associate the relevant visual and spatial features.

### 5.3.2 Meta-Update Phase

In the meta-update phase, shared meta-parameters of the MLP are updated to enable generalization across objects. This step balances the adaptation loss for the current object with the validation loss for unseen grasp data. Alternating between adaptation and meta-updates ensures a balance between specialization and generalization.

### 5.3.3 Context Extraction and Application

Once the adaptation and meta-update phases are complete, the deep kernel GP is used to extract context vectors for Bayesian optimization:

1. **Grasp Metric Mean:** For each object, compute the mean grasp success metric across all grasps.
2. **Feature Context Creation:** Use the MLP to process the grasp metric mean and produce a compact feature context vector with a dimensionality of (20,). This vector summarizes the key characteristics of how the object can be grasped.
3. **Using the Context Vector:** Treat the feature vector as context input to guide the BO process. This helps ensure the optimization is tailored to the specific grasping properties of the object.

By extracting and applying context in this way, the pipeline becomes highly flexible and efficient, making it adaptable to different grasping situations.

## 5.4 Contextual Analysis

The two losses depicted in the graphs, **adapt\_loss** and **valid\_loss**, have distinct roles in optimizing the model. **Adapt\_loss** is designed to fine-tune the model’s parameters, specifically the Gaussian Process (GP) parameters, to align with the training data for a particular object. It is primarily used during the adaptation phase to tailor the model for that specific object. Meanwhile, **valid\_loss** evaluates the model’s performance on a validation dataset composed of previously unseen data. This metric assesses the model’s generalization ability and helps update its meta-parameters based on the validation data.

Both graphs show a point of diminishing returns after step 150. Because neither adaptation loss nor validation loss continues to decrease beyond this point, it seems that the model has converged. Further training will not achieve the objective of minimizing these losses and may even risk overfitting. Hence, step 150 is also a suitable point to stop training. The losses are measured using the negative log-likelihood function,



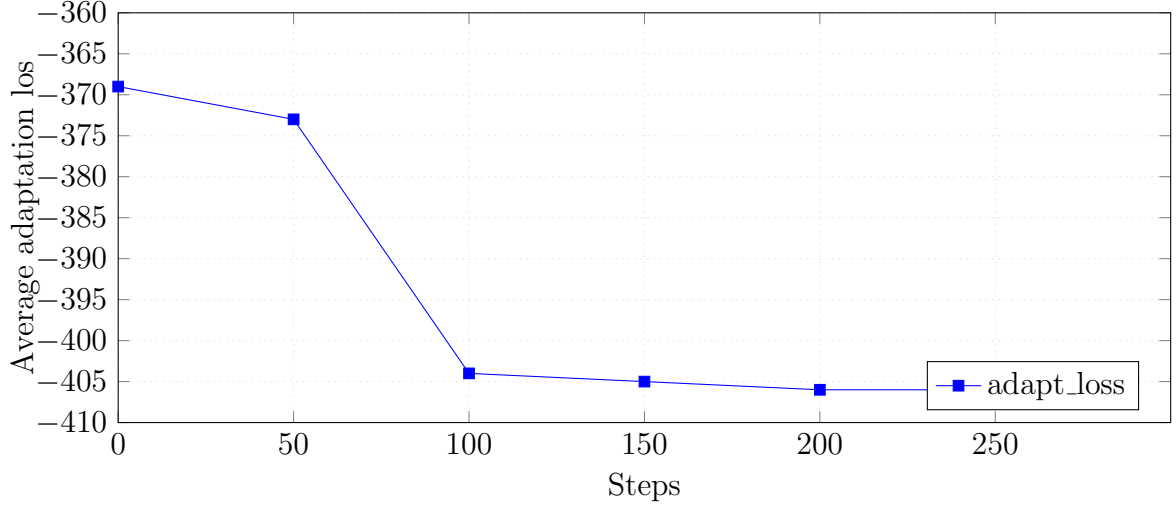


Figure 5.2: Average adaptation loss as a function of the number of steps..

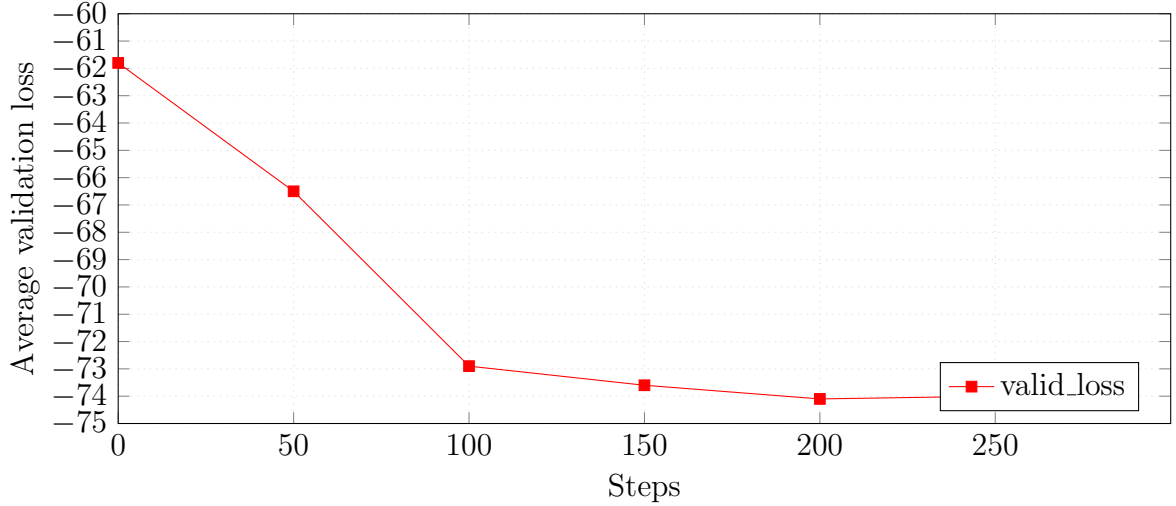


Figure 5.3: Average validation loss as a function of the number of steps.

a standard metric in regression tasks for probabilistic models. In this context, lower values signify better model performance, as they reflect closer alignment between the predicted distributions and actual data. Therefore, the goal of minimizing losses aligns with ensuring accurate, reliable predictions while optimizing computational resources.

This result aligns with the overall goal of keeping loss values small in order to have robust optimization and avoid unnecessary computations.

#### 5.4.1 Context-Aware Bayesian Optimization and Iterative Refinement

This work incorporates an iterative, context-aware Bayesian Optimization process to refine grasp configurations by leveraging accumulated contextual knowledge. This approach enhances convergence, improves robustness, and ensures efficient adaptation

across diverse objects.

After the initial optimization of an object, its grasp metrics and context vectors—derived from the grasp metric mean and visual embeddings—are added to a cumulative dataset. These vectors encapsulate object-specific characteristics, guiding future optimizations.

Once an object has been optimized, the grasp metrics and context vectors (the mean of the grasp metric and visual embeddings, respectively) are appended to this cumulative data set. These vectors capture object-specific features and assist in optimized future measurements.

At every iteration, the acquisition function (like Expected Improvement) is able to directly embed the information of the current object and the previous accumulated knowledge. This directs the optimization procedure towards areas of high potential in the searchspace, achieving a good trade-off between exploration and exploitation to increase the quality of the grasp.

The process continues until convergence, defined by stabilization of grasp quality metrics above a predefined threshold (e.g.,  $> 0.4$ ). Final results, including updated grasp configurations and context vectors, are stored to enrich the knowledge base for future tasks.

By iteratively integrating context and refining optimization, this method accelerates learning, minimizes redundant exploration, and achieves higher efficiency and adaptability in grasping tasks.

# Chapter 6

## Results: Impact of Context Integration

This chapter analyzes the effects of incorporating contextual information into the Bayesian Optimization (BO) process used to find grasp configurations. To assess this impact, several runs were carried out, comparing:

- **Non-context approach:** The BO model uses only data from its own iteration sequence for the object being optimized, without incorporating any extra features.
- **Context-aware approach:** In addition to the data collected in each iteration, contextual features generated by the Deep Kernel network are also used. These features capture latent information about each object, guiding the optimization more efficiently.

To ensure a fair comparison, both variants (with and without context) employ the same **Expected Improvement (EI)** acquisition function and are initialized with the same random seed in each run. Therefore, the only difference is the use of contextual features. Below, we show the optimization behavior for different objects, in the order they were introduced in the process.

### 6.1 Results Analysis and Observations

This section presents a comprehensive analysis of the grasp optimization results for the various objects tested during the experimental phase. The outcomes reflect the performance of the Bayesian optimization process, highlighting both convergence trends and object-specific behavior. Notably, the Bayesian optimization used a fixed seed to ensure reproducibility across runs, enabling a consistent and fair evaluation of the results.

The optimization process was conducted using the Expected Improvement (EI) acquisition function over 100 iterations. Each result indicates the gradual refinement

of grasp quality, with initial exploration yielding significant improvements that stabilize as the optimization progresses.

## Object-Specific Results and Observations

The grasp optimization performance varied across objects due to differences in their shapes, surface properties, and inherent graspability. For clarity, a summary of the maximum grasp quality achieved for each object is presented in Table 6.1.

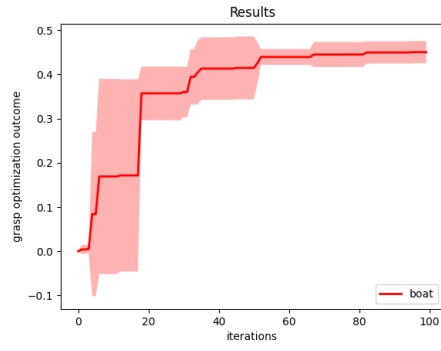
Tabla 6.1: Maximum Grasp Quality Achieved for Each Object

Object	Maximum Grasp Quality
Boat	0.45
Bottle	0.72
Duck	0.40
Charger	0.42
Mouse	0.35
Ice Cream	0.34
Trophy	0.40
Jar	0.50

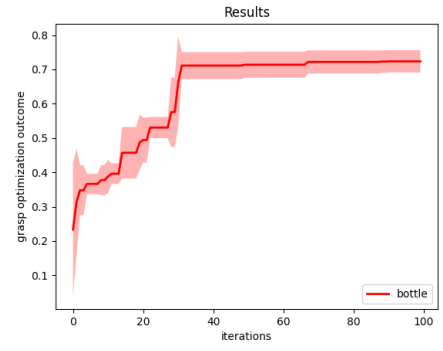
The boat object showed a very rapid early improvement, which plateaued around iteration 30. It shows that the algorithm quickly and effectively finds feasible configurations in the first few steps. Given that a cylinder has a large number of graspable regions, the bottle showed a sharp improvement in grasp quality early on. It converged in iteration 50. The unusual geometry of the duck made optimization difficult. Despite this, continuous improvements were obtained with grasp quality ultimately plateauing around iteration 40. The charger showed marginal improvement, achieving a clear plateau after around 50 iterations. This indicates the algorithm’s capability to learn similarly shaped objects and smaller objects. Well, keeping in mind the mice size and their flat surface and requiring cautious exploration to identify good grasp configurations, the mouse recognized more gradual optimization. We can see as with the bottle, that the ice cream object had rapid early progress, as the geometry such smooth and predictable geometry. Both processes converged at iteration 40. Trophy displays a monotonically increasing learning curve, stabilizing on grasp quality around iteration 50. This shows how flexible this method can be in handling complex and asymmetric shapes. Last but not least, the jar grasped with an absolute maximum quality score on all objects, with substantial gains after episode 30. Much of that success was owed to its regular geometry.

Rapid initial improvements were observed across most objects, demonstrating the efficiency of the Bayesian optimization approach. Simpler and more uniform

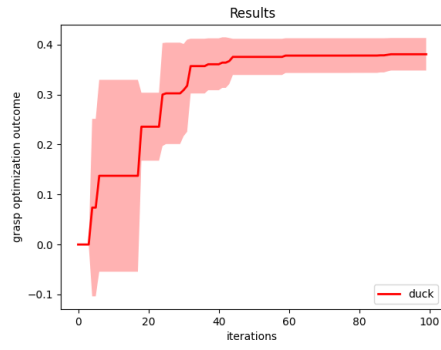
geometries, like the bottle and jar, reached higher maximum grasp quality values, while irregularly shaped objects, such as the duck and trophy, needed more iterations to stabilize due to their complex shapes. The accompanying figures visually present the optimization progress for each object, highlighting the grasp quality improvements over successive iterations.



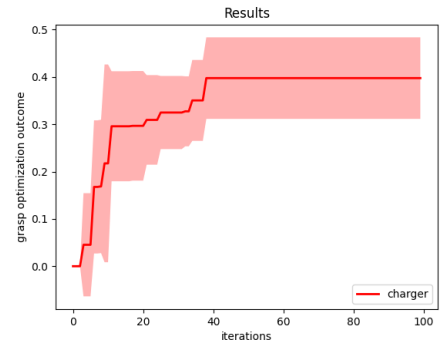
(a) Boat



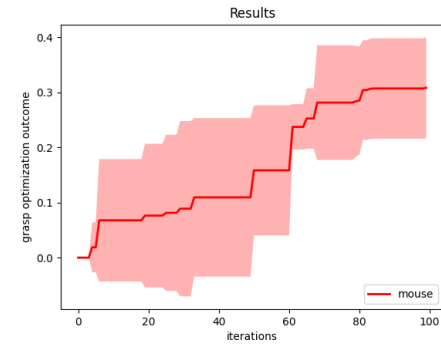
(b) Bottle



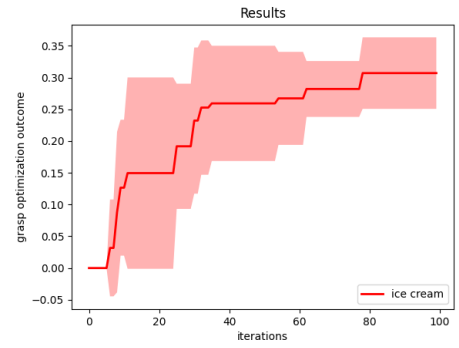
(c) Duck



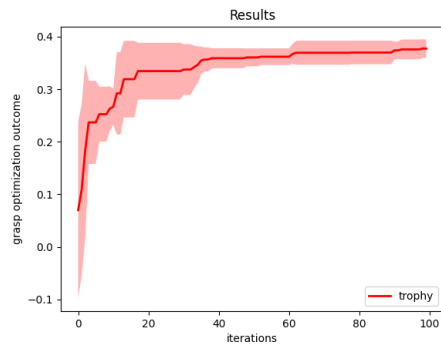
(d) Charger



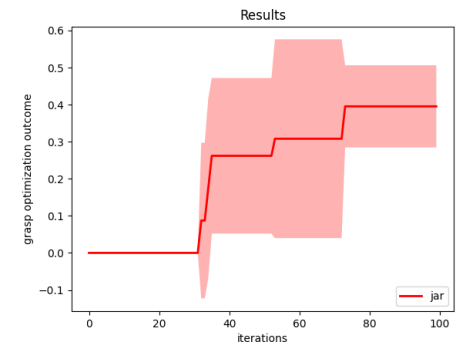
(e) Mouse



(f) Ice Cream



(g) Trophy



(h) Jar

Figure 6.1: Grasp optimization results for all objects. Each subfigure represents the grasp quality over iterations for a specific object.

## 6.2 Foundational Iterations: Establishing Context Through *boat* and *bottle*

The first optimization run starts with the *boat* object. At this point, there is no previous data or context, so the system depends entirely on the Deep Kernel’s data for *boat*. This makes *boat* the “base object” for starting the process.

After evaluating *boat*, we introduce the next object, *bottle*. Unlike *boat*, *bottle* can use contextual information carried over from *boat*’s data. This means the optimization combines the new data from *bottle* with what it learned from *boat*.

The comparison of *bottle* (shown in Figure 6.2) shows how grasp optimization improves as we run more iterations. Early on, both context-aware and context-free methods perform similarly because they lack enough prior data. However, as the process continues, the context-aware method achieves slightly higher final performance. This gain is small and might be affected by noise, but it still highlights a small advantage from adding context.

Overall, the *bottle* results demonstrate two main points:

1. Using context provides a small improvement at the later stages of optimization.
2. Because *bottle* is among the first objects tested, there is not a large history of previous objects, which limits the benefits of using context at this stage.

We chose *boat* and *bottle* as the first objects because they offer shapes that are both general and tend to perform well in context-free optimization runs. This makes them strong starting points for building a reliable baseline.

Even though *jar* also does well in context-free tests, its spherical shape does not help much in studying the impact of context. Spherical objects are simpler to optimize, so they do not provide the complexity needed to see a clear effect from additional contextual information.

On the other hand, *boat* and *bottle* have forms that balance complexity and general use. They are more representative of a variety of objects, allowing us to better explore how context can improve optimization when we move on to more diverse objects later.

By beginning with *boat* and *bottle*, the study ensures a well-grounded starting point, enhancing the reliability of conclusions drawn about the role of contextual information in the optimization process.

In the next set of experiments, we will reverse the order in which we test the objects. Instead of starting with *boat* and then moving on to *bottle*, we will do the opposite. By comparing the results of these two different sequences, we can see if the order of

objects changes how the context is used and whether it affects overall optimization performance.

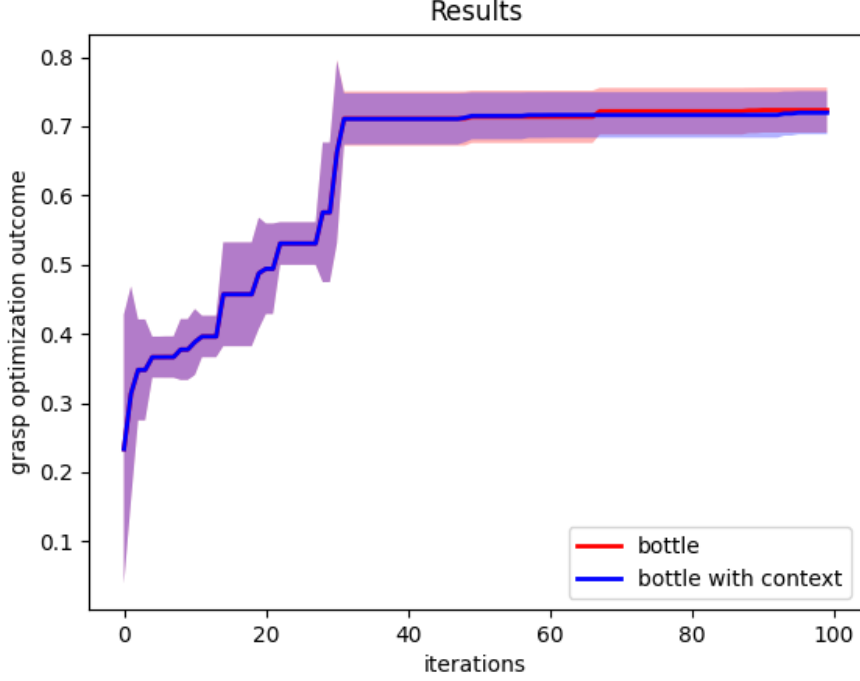


Figure 6.2: Optimization results for *bottle* without and with context. The two options are similar until reaching the end, where the context-aware approach generates marginally better results.

### 6.3 Iteration 1: Analysis of Object Optimization

The first experiments analyze how the optimization process occurs regarding a set of objects: *ice cream*, *jar*, *mouse*, *charger*, *duck*, and *trophy*. In this case, it benefits from the context provided earlier by *boat* and *bottle* objects. The present work provides a clearer picture of how using contextual information can enhance the use of Bayesian optimization for robotic grasping.

As for the object *ice cream*, the experiment demonstrates that the stability of the optimization can be improved using context, as evidenced by the smaller variance of the quality scores and a smoother optimization path. However, the resulting grasp quality is similar with or without context. For more straightforward objects, adding context does not really improve performance, but rather makes optimization more reliable.

For the *jar*, adding contextual information makes the process more stable, significantly cutting down on noise in the optimization curve. The clear and steady path of the context-aware method shows it can use prior knowledge effectively, leading



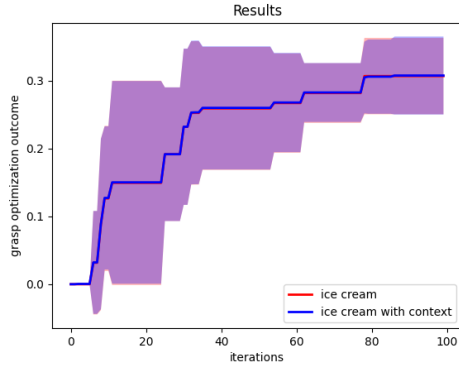
to smoother convergence and less variation.

The optimization results of the *mouse* showed that contextual data improved efficiency. The context-aware model achieves its maximum grip quality more quickly, lowering both computation and test costs. This is incredibly useful as the actual *mouse* is serpentine and more slippery, and therefore harder to hold on to.

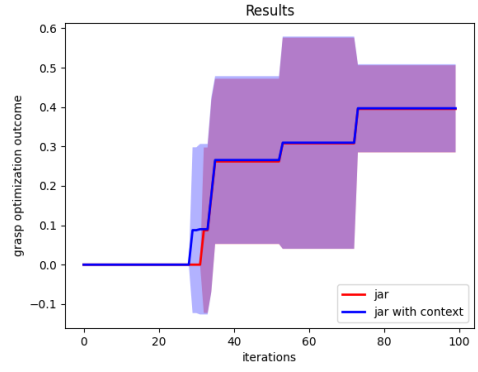
Interestingly enough, the *charger* object also provides more insight into this dual benefits of context for optimization. The context-aware approach not only converges more quickly but also maintains a steadier trajectory through the iterations. This is seen in the noticeably lower variability of the optimization curve compared to the method without context.

The results for *duck* highlight the strong advantages of using contextual information for complex, irregular objects. The context-aware method converges faster and achieves a better final grasp quality, outperforming the non-context approach after about 35 iterations. This emphasizes how important it is to accumulate contextual knowledge as the sequence advances and the system adapts to tougher shapes.

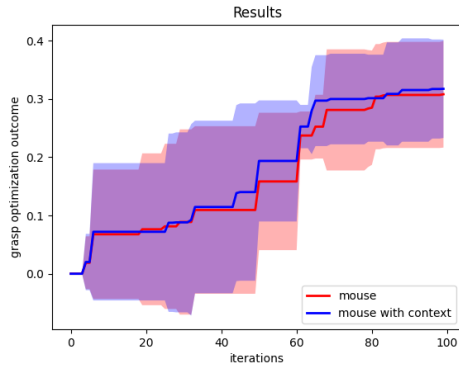
Finally, the *trophy* object gains significantly from the context-aware method, needing fewer iterations to stabilize and following a smoother path. While both methods end with a similar final grasp quality, the efficiency and stability of the context-aware method make it a more practical choice for real-world tasks.



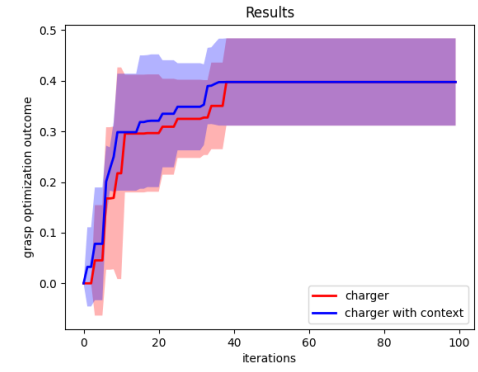
3<sup>o</sup>. Ice Cream



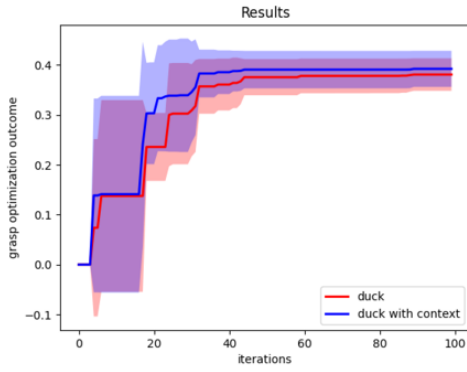
4<sup>o</sup>. Jar



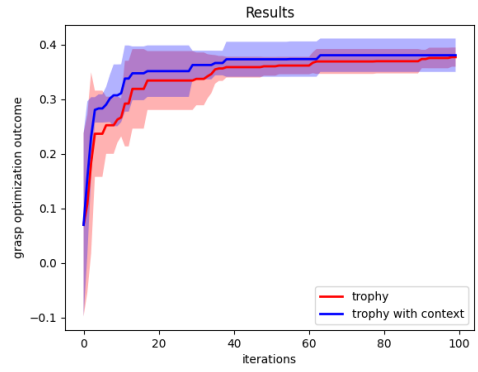
5<sup>o</sup>. Mouse



6<sup>o</sup>. Charger



7<sup>o</sup>. Duck



8<sup>o</sup>. Trophy

Figure 6.3: Optimization results for objects in Iteration 1: Comparison of context-aware and non-context methods for six objects: ice cream (3<sup>o</sup>), jar (4<sup>o</sup>), mouse (5<sup>o</sup>), charger (6<sup>o</sup>), duck (7<sup>o</sup>), and trophy (8<sup>o</sup>). Each plot highlights differences in stability, convergence speed, and final grasp quality.

Combined, the findings highlight the importance and strong benefits of using contextual information integrating Bayesian optimization on robotic grasping tasks. Both early-stage objects (e.g., *ice cream* and *jar*) also gain the benefits of stability and less noise, whereas mid-sequence objects (e.g., *mouse* and *charger*) converge faster and are more efficient. For complex objects like *duck* and *trophy*, the accumulated contextual knowledge improves robustness and final grasp quality. These findings

validate the effectiveness of context-aware Bayesian optimization in addressing diverse object geometries and complexities, paving the way for more efficient and adaptive robotic systems.

## 6.4 Iteration 2: Analysis of Object Optimization (Reversed Order)

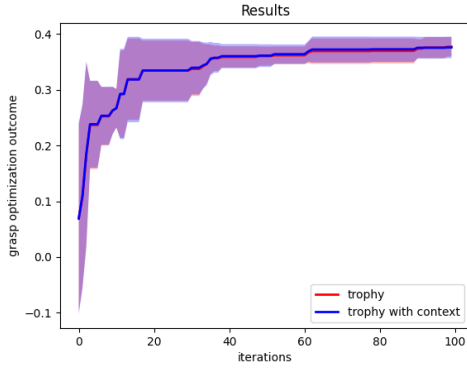
The second iteration analyzed the optimization process for the same objects as Iteration 1—*trophy*, *duck*, *charger*, *mouse*, *jar*, and *ice cream*—but in reversed order. This change in sequence aimed to explore whether the contextual impact becomes more significant as the optimization system accumulates experience with more objects. The results demonstrate distinct trends: for the earlier objects in the reversed sequence, the outcomes are nearly identical to those in Iteration 1, with context providing little advantage beyond noise reduction. However, for the later objects, the inclusion of context leads to significant improvements in convergence speed, stability, and final grasp quality.

The outputs for *trophy* and *duck*, the first objects in this reversed sequence, then, are nearly the same as we saw at step 1. This offers some stability compared to the non-context method, but the general patterns and final outcomes are similar. Implying that the system isn't yet presented with enough novel contextual information of an object to significantly improve optimization results on these early objects.

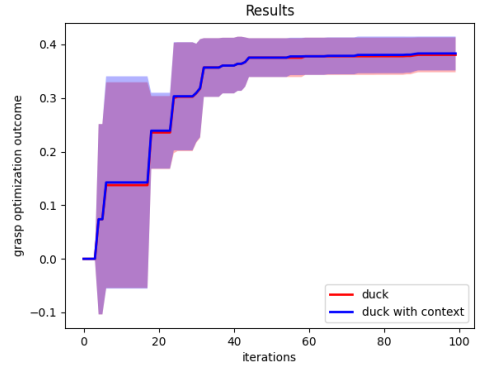
For *charger*, *mouse*, *jar*, and *ice cream*, the effect of context becomes more evident. By the time the system reaches *jar* and *ice cream*, the benefits of contextual optimization show up in faster convergence and smoother trajectories. The later objects in the reversed sequence clearly gain from the built-up contextual knowledge, achieving higher final grasp quality and needing fewer iterations to stabilize.

Comparing Iterations 1 and 2 reveals key insights into the role of object order in contextual optimization. For objects positioned early in the sequence, such as *trophy* and *duck*, the performance of context-aware and non-context methods is almost identical in both iterations. This is expected, as the system lacks sufficient accumulated contextual information at these early stages.

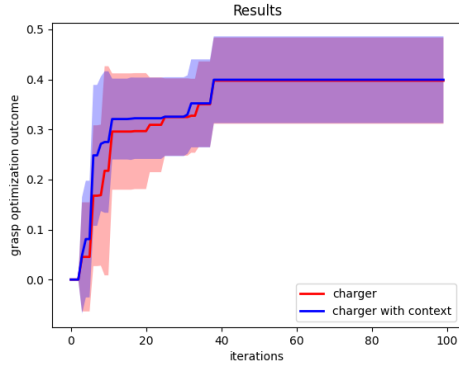
However, as the sequence progresses, the differences between the context-aware and non-context methods become increasingly significant. For mid-sequence objects like *charger* and *mouse*, the context-aware method in Iteration 2 achieves faster convergence and lower variability than in Iteration 1. By the time the system reaches the final objects, such as *jar* and *ice cream*, the improvements are particularly notable. In Iteration 2, the context-aware approach achieves higher final grasp quality with fewer



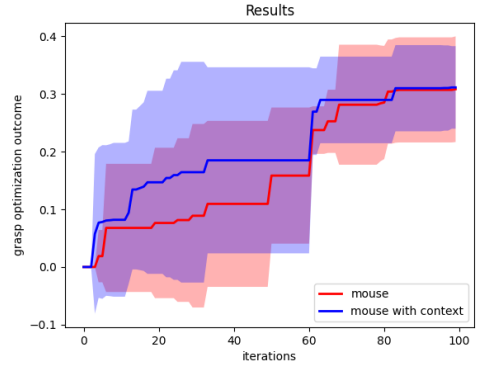
3<sup>o</sup>. Trophy



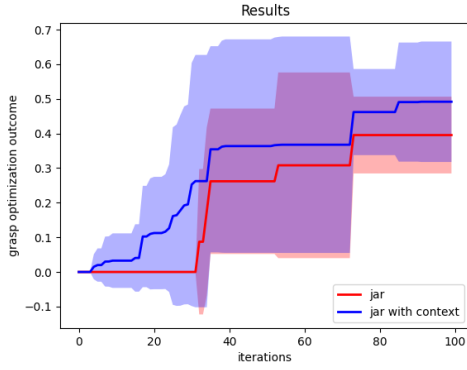
4<sup>o</sup>. Duck



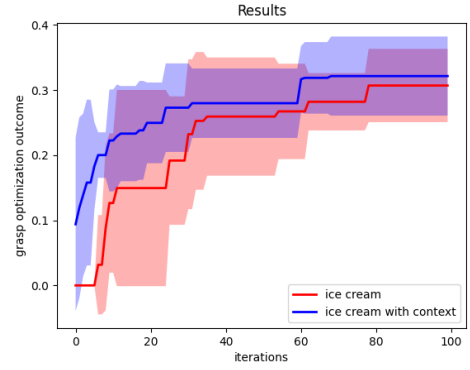
5<sup>o</sup>. Charger



6<sup>o</sup>. Mouse



7<sup>o</sup>. Jar



8<sup>o</sup>. Ice Cream

Figure 6.4: Optimization results for objects in Iteration 2 (reversed order): *trophy* (3<sup>o</sup>), *duck* (4<sup>o</sup>), *charger* (5<sup>o</sup>), *mouse* (6<sup>o</sup>), *jar* (7<sup>o</sup>), and *ice cream* (8<sup>o</sup>). The earlier objects (*trophy*, *duck*) show minimal differences, while the later objects (*mouse*, *jar*, *ice cream*) exhibit significant improvements with the context-aware method.

iterations, indicating that the accumulation of contextual knowledge over time enhances the system’s ability to generalize and adapt to new objects.

The reversed order in Iteration 2 highlights the importance of cumulative context in Bayesian optimization. While early-stage objects see minimal benefits from context, the impact increases as the number of objects the system handles increases. That

observation indicates that the utility of contextual optimization is magnetic towards the amount of initial data. Therefore, making it unique for complicated or unique items that appear in the end. The new results also reaffirm that context-aware approaches have the potential to enhance the efficiency and robustness of robotic grasping strategies, as they can adapt to different orders of object interactions that may occur in dynamic environments.

## 6.5 Overall Discussion of the Results

Taken together, the findings across different objects highlight several key conclusions:

- **Stability and noise reduction:** From the earliest objects, the context-aware version tends to exhibit smoother curves and narrower confidence bands, indicating lower variance in predictions.
- **Faster convergence:** As more objects are introduced, the context-aware approach usually finds optimal grasp configurations with fewer iterations.
- **Slight but consistent final gains:** While the differences in the maximum grasp quality are not always huge (especially for less complex objects or those introduced early on), we do see that for objects like *duck* or *charger*, the context-aware method attains better final values and/or does so more consistently.
- **Dependency on history:** The benefit of using context becomes more evident as the system processes a larger number of objects, suggesting that the Deep Kernel network has more useful information to differentiate and guide searches in subsequent objects.
- **Importance of object order:** The results show that object order during processing is key for the system’s performance. Preliminary objects set the background context, thus choosing early objects that generalize well, and have good outcomes is critical. This is, however, only possible if a solid base can be built which guarantees that the queries and the contextual features that have been accumulated offer meaningful advice on which optimization problem to solve next. Variable features may also improve efficiency in later (and more complex) objects, but poorly chosen starting objects can restrict system adoption and efficiency moving forward in the sequence.

In summary, the results support the hypothesis that contextual information—implemented through features learned by the Deep Kernel network—can help the Bayesian optimization of grasps become more efficient (fewer iterations) and more robust (more stable trajectories). However, the improvements in terms of the final grasp quality are not always dramatic and may vary depending on each object’s complexity, the system’s maturity (i.e., how many objects have been analyzed so far), and the representativeness and quality of the initial objects used to build the contextual foundation.

### 6.5.1 Future Directions and Practical Applications

This work establishes a solid foundation for advancing robotic grasping through context-aware Bayesian optimization, and several promising directions for future research and implementation arise. Expanding the dataset with more diverse objects, including complex, deformable, or edge-case geometries, would provide a broader validation of the system’s adaptability. The strategic selection of initial objects is also critical—choosing foundational items with diverse and representative geometries can enhance the generalizability of contextual features and improve performance on subsequent objects.

Dynamic learning methods, in which the system continuously deepens its contextual knowledge base with novel objects stepwise, would ensure both an efficient and an optimization process. Additionally, integrating the framework with actual robotic platforms would further evaluate its robustness with respect to realistic constraints, including sensor noise and imperfect scenarios. Another exciting direction is multi-robot coordination where robots share their contextual knowledges and jointly optimize their tasks in complex environments.

Finally, enhancing the computational efficiency of the system to allow real-time applications would enable its use in industrial automation, warehousing, and domestic robotics. Future work can build on these directions for increasing the adaptability, scalability, and practical use of context-aware robotic systems, aiding further developments in autonomous object manipulation.

# Chapter 7

## Conclusions

This research offers a thorough examination of robotic grasping, using a humanoid robotic hand as a testing platform and integrating advanced techniques such as DINO self-supervised learning and Bayesian Optimization. By incorporating context-aware strategies, we demonstrate notable progress in robotic manipulation, tackling key challenges in both precision and adaptability. The approach and findings can also be extrapolated to other robotic systems with similar capabilities.

The main conclusions are basically that the benefits of contextual data are increasingly noticeable as the process of optimization progresses. The improvements, though, done in the early stages are pretty low and can at best be explained by noise given the slight contextual information therein. However, as the system learns from increasing contextual data, the clear gain in efficiency is observed clearly. For later-encountered objects in the process, fewer observation optimization iterations (fewer steps on the X-axis) are required to obtain similar or slightly improved grasping performance (Y-axis).

This behavior highlights how gradually building up context leads to short-circuiting reasoning. At later stages, the system converges faster and demonstrates consistent albeit moderate improvements in grasp performance, indicating that the advantages are no longer attributable merely to noise. This allows the subsequent optimization process to be guided by the context over the gathered data, ensuring that a single integrated system learns from what it already knew about its tasks and generalizes from it to new objects.

In addition, the results from the two different object sequences analyzed (Iterations 1 and 2) emphasize the importance of object order in the optimization process. Objects introduced early in the sequence play a critical role in defining the contextual foundation. Initial objects that generalize well and exhibit good outcomes significantly enhance the effectiveness of subsequent optimization tasks. Conversely, poorly chosen initial objects with limited representativeness can hinder the system's ability to

adapt efficiently to more complex objects later in the process. This underscores the importance of strategic selection of early objects to maximize the benefits of context-aware optimization.

Such results collectively illustrate the potential for unifying self-supervised learning, deep kernel representations, and Bayesian Optimization. The method leverages contextual information to achieve significant computational and experimental savings with comparable or improved performance metrics.

To sum up, our results show indeed that, while reward shaping can be helpful in the very early stages, restraining specifically from context information, context-aware approaches bring many advantages during later (and in real situations, most of the time stressful) tasks, all of which are essential traits behind quicker, safer, and smarter robotic systems. This research makes an important contribution to robotics by emphasizing how cumulative learning and context-aware optimization pave the way for intelligent and efficient grasping.



# Chapter 8

## Bibliography

- [1] Ignacio Herrera Seara, Juan García-Lechuz Sierra, Javier García-Barcos, and Rubén Martínez-Cantín. Optimización bayesiana multisolución para la exploración eficiente de agarres robóticos. In *XLIII Jornadas de Automática: libro de actas*, pages 714–720. Universidade da Coruña, 2022.
- [2] M. Caron, H. Touvron, I. Misra, H. Jégou, J. Mairal, P. Bojanowski, and A. Joulin. Dino: Emerging properties in self-supervised vision transformers. *arXiv preprint arXiv:2205.02708*, 2022.
- [3] David J. Montana. The kinematics of contact and grasp. *The International Journal of Robotics Research*, 7(3):17–32, 1995.
- [4] Mark R. Cutkosky. On grasp choice, grasp models, and the design of hands for manufacturing tasks. *IEEE Transactions on Robotics and Automation*, 5(3):269–279, 1989.
- [5] Marco Santello, Martha Flanders, and John F. Soechting. Postural hand synergies for tool use. *The Journal of Neuroscience*, 18(23):10105–10115, 1998.
- [6] John J. Craig. *Introduction to Robotics: Mechanics and Control*. Pearson Prentice Hall, 3rd edition, 2005.
- [7] Bruno Siciliano, Lorenzo Sciavicco, Giuseppe Villani, and Giuseppe Oriolo. *Robotics: Modelling, Planning and Control*. Springer, 2010.
- [8] Oussama Khatib. A unified approach for motion and force control of robot manipulators: The operational space formulation. *IEEE Journal on Robotics and Automation*, 3(1):43–53, 1987.

- [9] A.T. Miller and P.K. Allen. Examples of 3d grasp quality computations. In *Proceedings 1999 IEEE International Conference on Robotics and Automation*, volume 2, pages 1240–1246, 1999.
- [10] João Castanheira, Pedro Vicente, Ruben Martinez-Cantin, Lorenzo Jamone, and Alexandre Bernardino. Finding safe 3d robot grasps through efficient haptic exploration with unscented bayesian optimization and collision penalty. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1643–1648, 2018.
- [11] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. Dinov2: Learning robust visual features without supervision. *Meta AI Research*, 2024. Reviewed on OpenReview.
- [12] Ricardo Vilalta and Youssef Drissi. A perspective view and survey of meta-learning. *Artificial Intelligence Review*, 18(2):77–95, 2002.
- [13] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning*, pages 1126–1135, 2017.
- [14] Timothy M. Hospedales, Antreas Antoniou, Paul Micaelli, and Amos J. Storkey. Meta-learning in neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(9):5149–5169, 2021.
- [15] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. In *5th International Conference on Learning Representations*, 2017.
- [16] Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W. Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando de Freitas. Learning to learn by gradient descent by gradient descent. In *Advances in Neural Information Processing Systems*, pages 3981–3989, 2016.
- [17] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando De Freitas. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2016.

- [18] Donald R Jones, Matthias Schonlau, and William J Welch. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13(4):455–492, 1998.
- [19] Eric Brochu, Vlad M Cora, and Nando De Freitas. A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv preprint arXiv:1012.2599*, 2010.
- [20] Carl Edward Rasmussen and Christopher KI Williams. *Gaussian processes for machine learning*. MIT Press, 2004.
- [21] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems*, pages 2951–2959, 2012.
- [22] Jonas Mockus, Vytautas Tiesis, and Antanas Zilinskas. The application of bayesian methods for seeking the extremum. *Towards Global Optimization*, 2:117–129, 1975.
- [23] Harold J Kushner. A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise. *Journal of Basic Engineering*, 86(1):97–106, 1964.
- [24] Niranjana Srinivas, Andreas Krause, Sham Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. *Proceedings of the IEEE*, pages 1015–1022, 2010.
- [25] Kevin Swersky, Jasper Snoek, and Ryan P. Adams. Multi-task bayesian optimization. In *Advances in Neural Information Processing Systems*, pages 2004–2012, 2013.
- [26] Wenlin Chen, Austin Tripp, and José Miguel Hernández-Lobato. Meta-learning adaptive deep kernel gaussian processes for molecular property prediction. *arXiv preprint arXiv:2205.02708*, 2022.
- [27] GPJax Developers. Deep kernel learning. <https://docs.jaxgaussianprocesses.com/examples/deep> 2025 – 01 – 24.
- [28] Javier Garcia-Barcos and Ruben Martinez-Cantin. Fully distributed bayesian optimization with stochastic policies. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages

2357–2363. International Joint Conferences on Artificial Intelligence Organization, 7 2019.

- [29] Ruben Martinez-Cantin, Nando de Freitas, Eric Brochu, José A. Castellanos, and Arnaud Doucet. A bayesian exploration-exploitation approach for optimal online sensing and planning with a visually guided mobile robot. *Autonomous Robots*, 27(2):93–103, 2009.
- [30] Nikolaus Vahrenkamp, Manfred Kröhnert, Stefan Ulbrich, Tamim Asfour, Giorgio Metta, Rüdiger Dillmann, and Giulio Sandini. Simox: A robotics toolbox for simulation, motion and grasp planning. In *Proceedings of the International Conference on Intelligent Autonomous Systems (IAS)*, pages 585–594, 2012.
- [31] G. Metta, L. Natale, and F. Nori. The icub humanoid robot: An open-systems platform for research in cognitive development. *Neural Networks*, 23(8):1125–1134, 2010.
- [32] Ruben Martinez-Cantin. Bayesopt: A bayesian optimization library. <https://rmcantin.github.io/bayesopt/>, 2023. Accessed: 2025-01-04.
- [33] N. G. Tsagarakis, G. Metta, and G. Sandini. icub: The design and realization of an open humanoid platform for cognitive and neuroscience research. *Advanced Robotics*, 21(10):1151–1175, 2007.
- [34] G. Sandini, G. Metta, and D. Vernon. Robotics and neuroscience. *Current Biology*, 17(15):R585–R590, 2007.
- [35] S. Denei, F. Nori, and G. Metta. Contact and tactile sensing for robot manipulation. *Frontiers in Neurorobotics*, 9, 2015.
- [36] G. Metta, F. Nori, and L. Natale. Sensorimotor integration and learning in humanoid robots: The icub approach. *Robotics and Autonomous Systems*, 55(2):1–19, 2008.
- [37] Christopher KI Williams and Carl Edward Rasmussen. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

# List of Figures

1.1	Modern humanoid robotic hand showcasing advanced design for grasping and manipulation. . . . .	2
1.2	Gantt chart illustrating the timeline and structure of the project, showing the distribution of tasks across several months. . . . .	5
2.1	An example of a robotic hand performing a grasp on an object. The image highlights the contact points and alignment between the hand and the object. . . . .	8
2.2	The classification of robotic grasps: 15 types of commonly used grasping modes. This diagram highlights the diversity of grasping strategies that robotic hands can adopt. . . . .	10
2.3	Visualization of DINOv2’s capabilities: The framework processes images (left) to produce the segmentation maps (middle) and to apply visual overlays (right). These illustrate how high-quality representations learned in DINO can make robust, spatially consistent features to solve downstream tasks, such as grasp prediction in robotic systems. Adapted from [11]. . . . .	12
2.4	Diagram of Meta-Learning Architecture: A visual representation illustrating how a meta-learning model processes multiple tasks to learn general patterns that facilitate adaptation to new tasks. Such diagrams help in understanding the structure and internal functioning of meta-learning models. . . . .	14
2.5	Schematic representation of Adaptive Deep Kernel Fitting with Implicit Function Theorem (ADKF-IFT), illustrating the meta-learning process [26]. . . . .	21
3.1	Training pipeline: Visual embeddings are extracted using DINO from 8 images of the objects, while metrics logs from Bayesian optimization (EI, LCB, cAopt) are integrated to generate a deep kernel. . . . .	23
3.2	Context-Based Object Optimization Pipeline scheme . . . . .	26

4.1	Visual representations of the objects (Jar, Bottle, Duck, Boat, Trophy, Mouse, Charger, and Ice Cream) processed with Dino. . . . .	28
4.2	The iCub Robot. . . . .	30
5.1	Examples of good and poor grasps for different objects. Good grasps (quality $> 0.3$ ) are on the left, while poor grasps (quality $< 0.3$ ) are on the right. The objective is to achieve high-quality grasps (target $> 0.4$ ) across all objects. . . . .	36
5.2	Average adaptation loss as a function of the number of steps.. . . .	41
5.3	Average validation loss as a function of the number of steps. . . . .	41
6.1	Grasp optimization results for all objects. Each subfigure represents the grasp quality over iterations for a specific object. . . . .	46
6.2	Optimization results for <i>bottle</i> without and with context. The two options are similar until reaching the end, where the context-aware approach generates marginally better results. . . . .	48
6.3	Optimization results for objects in Iteration 1: Comparison of context-aware and non-context methods for six objects: ice cream ( $3^\circ$ ), jar ( $4^\circ$ ), mouse ( $5^\circ$ ), charger ( $6^\circ$ ), duck ( $7^\circ$ ), and trophy ( $8^\circ$ ). Each plot highlights differences in stability, convergence speed, and final grasp quality. . . . .	50
6.4	Optimization results for objects in Iteration 2 (reversed order): <i>trophy</i> ( $3^\circ$ ), <i>duck</i> ( $4^\circ$ ), <i>charger</i> ( $5^\circ$ ), <i>mouse</i> ( $6^\circ$ ), <i>jar</i> ( $7^\circ$ ), and <i>ice cream</i> ( $8^\circ$ ). The earlier objects ( <i>trophy</i> , <i>duck</i> ) show minimal differences, while the later objects ( <i>mouse</i> , <i>jar</i> , <i>ice cream</i> ) exhibit significant improvements with the context-aware method. . . . .	52

# List of Tables

4.1	Degrees of Freedom (DoF) of the iCub Robot. . . . .	30
6.1	Maximum Grasp Quality Achieved for Each Object . . . . .	44