Multi-Purpose Mechanical Arm

Group 2

Gabriel Forsberg – 200012044012
Per-Ola Gradin – 199803293233
Andreas Munck af Rosenschöld – 200303147672
Oscar Wir – 200112054416
Titouan Le Hong – 20030223T436

Chalmers University of Technology

November 17, 2025

Abstract

This report presents the design and implementation of a low-cost, multi-purpose robotic arm intended to approximate the dexterity and force awareness of a human arm while remaining compatible with rapid prototyping. Building on the Hubert platform from the Humanoid Robotics Lab at Chalmers, the arm's mechanical structure was redesigned with interchangeable 3D-printed links and standard metal U- and servo brackets, and its actuation was upgraded with higher-torque servos to increase reach and payload. A passive torque-assistance system combining a counterweight and springs was added to reduce joint loads. The end effector is an underactuated, tendon-driven hand using a single tendon routed through pulley triangles to drive three fingers, each equipped with thin-film force-sensing resistors (SEN0294) for grip-force feedback. Perception is provided by an end-effector-mounted camera and a and an open-vocabulary YOLOE-11L model, combined with monocular localization under a known plane assumption. Motion control relies on an Arduino Uno receiving PWM targets over serial and on an inverse-kinematics neural network trained in an autoencoder-like manner using the arm's forward kinematics to enforce both end-effector position and a downward grasping orientation. The system supports both autonomous pick-and-place and teleoperation via a PS5 controller. Experiments show that the trained IK can reconstruct reachable end-effector poses with sub-millimetre error in simulation and that the arm can detect and move simple tabletop objects within a limited 135° workspace. However, friction in the improvised pulley routing, camera placement, servo overload (especially at the base and elbow), the lack of closed-loop use of the force sensors, and the time-based control of the continuous-rotation gripper currently limit precision, durability, and generality. The results nonetheless demonstrate that a modular, vision-guided, underactuated arm with learning-based IK can be realized using mostly off-the-shelf and 3D-printed components, and they outline clear avenues for improvement in sensing, actuation, and grasp control.

Contents

1 Introduction

One of the main challenges in creating a robotic arm that resembles the human arm in both form and function is replicating its high dexterity and sensitivity to external forces. State-of-the-art approaches include using force/torque transducers to measure loads applied to a rigid structure (such as the tip of a finger) and converting the resulting mechanical strain into electrical signals, or embedding small magnets within a soft elastomer and using Hall sensors to detect changes in the surrounding magnetic field [1, 2]. While proven effective, these methods are often either prohibitively expensive or difficult to integrate in designs where space, weight, or material flexibility are limited. The goal of this project was therefore to design and construct an arm capable of sensing both the applied and exerted forces, while maintaining moderate to high dexterity and keeping overall costs low.

To achieve this, a combination of low-cost commercial and 3D-printed components was used, together with computer vision models such as the Ultralytics YOLOE model. The design is built upon the *Hubert* robot architecture (see Figure 1) developed by the Humanoid Robotics Lab at Chalmers University, extending its range, strength, and overall versatility by upgrading the servos and links used, and by incorporating open-source designs from the Yale OpenHand Project [3] as well as custom parts with inspiration from the OpenHand Project.

1.1 Purpose

The purpose of this project is to design and construct a robotic arm that resembles the dexterity and sensitivity of a human hand, enabling effective manipulation in a variety of practical scenarios. By leveraging low-cost materials and accessible technologies, this robotic arm aims to promote hands-on experimentation and prototyping in robotics.

1.2 Aim

To design, prototype, and test a multi-purpose robotic arm, based on the Hubert platform and capable of flexible grasping, real-time sensor feedback, and precise adaptive manipulation.

1.3 Scope

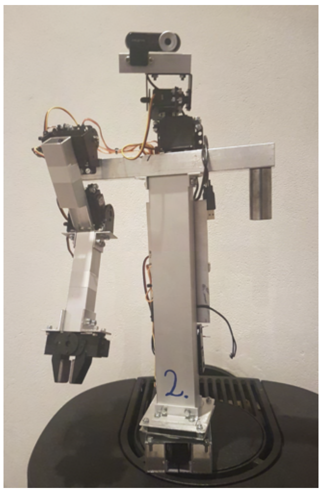
The scope of this project includes:

1. **Design and Development:** Creating a robotic arm based on the Hubert platform, focusing on mechanical components and joint configurations.
2. **Material Selection:** Using low-cost commercial parts and 3D-printed components for affordability and ease of assembly.
3. **Sensor Integration:** Incorporating sensors to provide real-time feedback on force and dexterity for adaptable manipulation.
4. **Software Development:** Utilizing libraries, such as OpenCV, Ultralytics and PyTorch for visual processing and control.
5. **Testing and Iteration:** Evaluating performance in various scenarios and refining the design based on feedback.

2 Hardware Design

The complete robotic arm can be divided into three main subsystems: the arm links and joints, the pulley-tendon transmission system, and the end-effector fingers. Each subsystem presents its own set of design considerations, mechanical constraints, and performance trade-offs. Together, they define the overall capabilities of the robot in terms of reach, payload capacity, and manipulation dexterity.

The design of the new arm builds upon the original Hubert architecture, addressing limitations in range of motion, load handling, and grasping efficiency. Figure 1 shows a comparison between the original Hubert robot(left) and the fully assembled redesigned arm (right).


(a) Original Hubert robot.

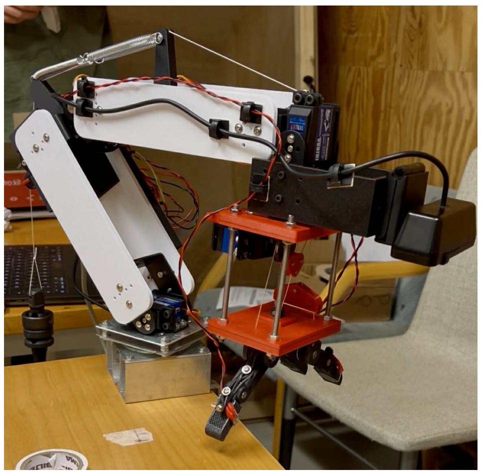

(b) Redesigned robotic arm with updated joints, tendons, and end-effector.

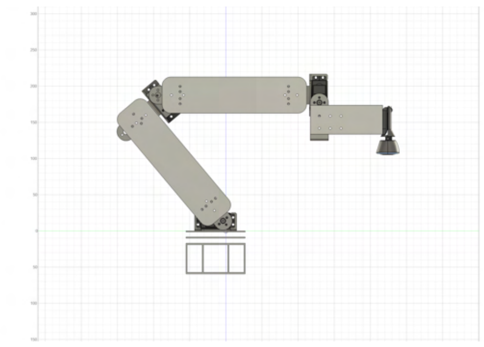
Figure 1: Comparison between the original Hubert robot (a) and the redesigned arm assembly (b).

The redesigned arm reuses the base plate from the Hubert robot with its servo for rotating the arm, and reuses the electrical system including the Arduino UNO microcontroller and the breadboard used for connecting servos and sensors. The arm has a lower link with a pivotal joint in connection to the base plate, and another pivotal joint in connection to the upper arm link. The hand is mounted to the end of the upper link with another pivotal joint, with a camera mounted in a U-bracket with a fixed position relative to the hand. The hand consists of an underactuated pulley-tendon system driven by a single servo controlling the three fingers, with force-sensitive resistors on each finger for sensory feedback on gripping pressure.

2.1 Arm Links and Joints

The design of the Hubert robot imposes several constraints on the types of objects it can manipulate and the range of motion it can achieve. In its initial configuration, the gripper was only able to apply sufficient force to grasp very light objects, and the reachable workspace was limited to a narrow operating range. The first design improvement was therefore to integrate more powerful servomotors in all but the rotational servo mounted at the base, which experiences comparatively lower torque demands. This upgrade not only provided greater control over the applied grasping force and lifting capability but also enabled the use of longer link segments, thereby increasing the arm's reach and overall mobility. The added, more powerful, servos used for the arm joints were Injora INJS035-270 digital servos with 270° rotation and 35 kg·cm torque rating, and the re-used base servo was a Hitec HS-422 standard servo with 180° rotation and a 4.1 kg·cm torque rating.

A second major design decision was to make the links between joints interchangeable. This modular approach allowed the arm's proportions to be quickly reconfigured to address issues such as insufficient reach, load imbalance, or changes in servo performance resulting from other design adjustments during prototyping. In Figure 2, two illustrations of two different link configurations are shown, equidistant links (left) and a short shoulder link with a long elbow link (right).


(a) Used configuration.

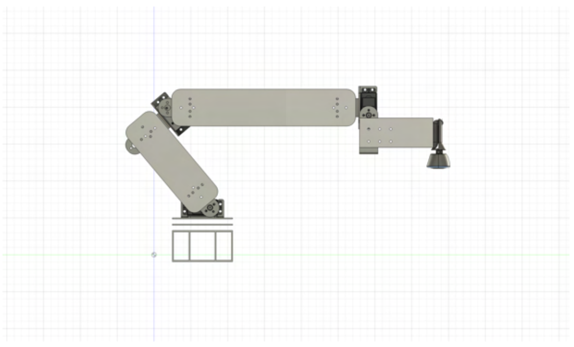

(b) Alternative configuration with longer reach.

Figure 2: Comparison of two link configurations for the Hubert robotic arm. (a) The configuration used in the final prototype with equal-length links. (b) An alternative configuration featuring a shorter shoulder link and longer elbow link, offering increased reach at the cost of reduced lifting strength and higher torque demand on the elbow joint.

Finally, to further support rapid prototyping and minimize resource use, the joints were simplified to consist solely of servo mounting brackets and standard U-brackets. This design choice reduced complexity, improved maintainability, and made it easier to iterate on the mechanical structure during development. In addition, the metal brackets provided sufficient structural support at all major stress points of the arm.

A late-stage addition to the design was the implementation of a string-based counterweight system to reduce the torque experienced by the elbow servo under heavy loads. This passive assistance mechanism helped distribute the mechanical load more evenly across the arm structure, improving both lifting performance and servo longevity. The weight required to alleviate the stress on the elbow servo however proved to be too heavy for the shoulder servo below. A lighter weight and additional springs were thus mounted to achieve the final balance needed to move as intended in all directions (see Figure 3).

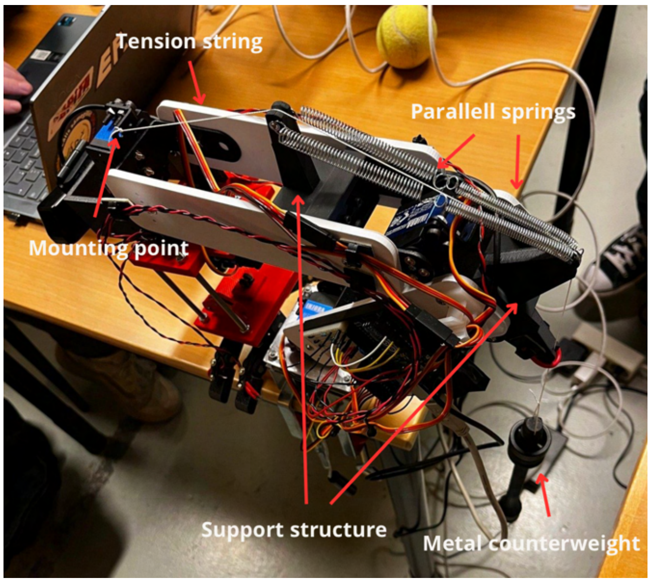
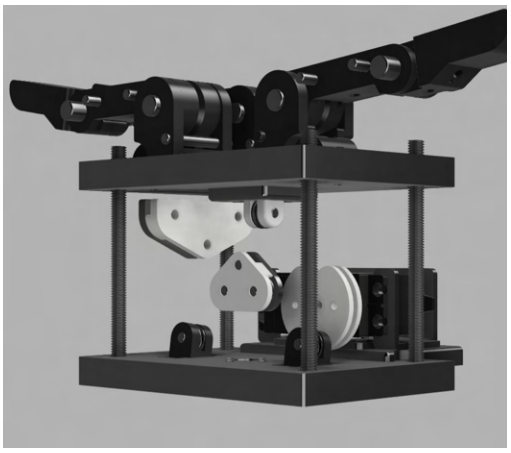


Figure 3: An overview of the combined spring and counterweight torque assistance system.

2.2 Pulley-Tendon System

The end-effector design is centered around a pulley-tendon transmission system that enables under-actuated and adaptive grasping. This system, which is the main body of the hand developed for this project, is designed with inspiration drawn from the open-source Yale OpenHand Project [3], specifically the design from the model T version. The system is driven by a single continuous rotation servo, with an attached main tendon that is routed through a system of pulleys and pulley triangles designed to distribute force equally among the attached fingers. An overview of the design of the pulley-tendon system is shown in Figure 4, rendered from the design in Autodesk Fusion, together with an image showing the final assembly of the hand.


(a) 3D model of the pulley-tendon system.

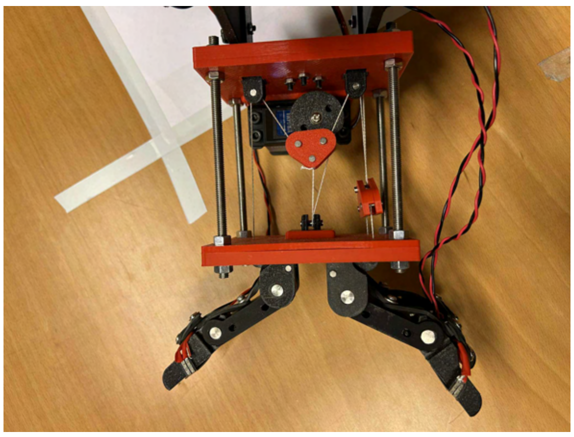

(b) Final assembly of the hand.

Figure 4: The pulley-tendon system consisting of a servo, pulleys and pulley triangles placed between two 3D printed plates.

The servo used to drive the pulley-tendon system is an Injora INJS035-360 continuous rotation servo with 35 kg·cm torque, mounted in a standard servo bracket for modularity and ease of replacement. The servo drives a main tendon, or fishing line in this case, by rotating a disk which the tendon is attached to. A continuous rotation servo was chosen to ensure a capacity of rotation that is sufficient to fully open and close the fingers, and also allows for different size of disks allowing for a stronger torque using smaller disks if needed.

In Figure 5, a more detailed view of the pulley-tendon system and the tendon routing can be seen. The pulley-tendon architecture allows the three fingers to close simultaneously around objects of varying geometry without the need for individual actuation. This approach simplifies control while maintaining dexterity, with effective distribution of forces and adaptive grasping of objects with various shapes.

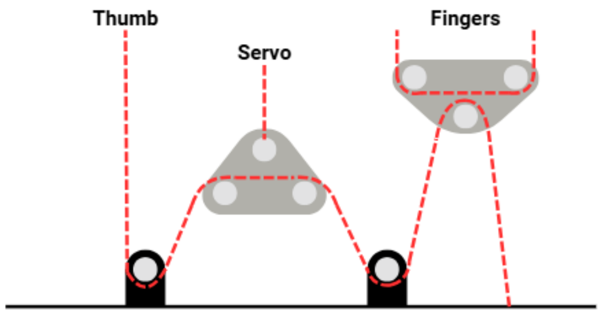


Figure 5: Schematic view of the tendon routing through the pulley-tendon system. The red dashed line shows the tendons, and the grey cylinders represent the stainless steel axles which act as pulleys in this system.

The servo drives a main tendon routed to a pulley triangle, balancing loads between the "thumb" and fingers on opposite sides of the hand. On the side with two fingers, an additional pulley triangle is placed to balance force between the two fingers. To achieve equal force between the thumb and the two fingers, the tendon connected to the finger-side pulley triangle is routed down to a fixed attachment to the base plate, leading to a doubling of the force. This was added during the assembly process, where it became clear that the tendon to the thumb had double the force of the fingers on the opposite side.

Initially, the tendons were designed to be routed through nylon pulleys, but due to unavailability of parts they were ultimately routed directly onto the stainless steel axles initially designed to hold the nylon pulleys in place. This potentially led to an increased friction in the system, and even after doubling the force to the pulley triangle connected to the two fingers, they still had more resistance when closing compared to the thumb. Although this compromise introduced higher friction and a slight imbalance between fingers, the system remained functional and demonstrated stable grasping.

2.3 End-effector Fingers

As with the pulley-tendon system, the design of the end-effector fingers draws inspiration from the Yale OpenHand project [3]. The fingers are designed to resemble human fingers, with one joint at the base of the fingers and one joint close to their middle. Each finger is composed of two 3D printed phalanges, and a 3D printed attachment to the rest of the hand. The pivot joints in the fingers are held together by stainless steel axles mounted within the 3D printed parts.

The fingers are controlled by the aforementioned pulley-tendon systems, with a tendon routed through each finger controlling their flexion, and rubber bands mounted to screws on the back of the fingers giving automatic extension when releasing the grip (see Figure 8b). An overview of the finger design and the tendon routing through them can be seen in Figure 6.

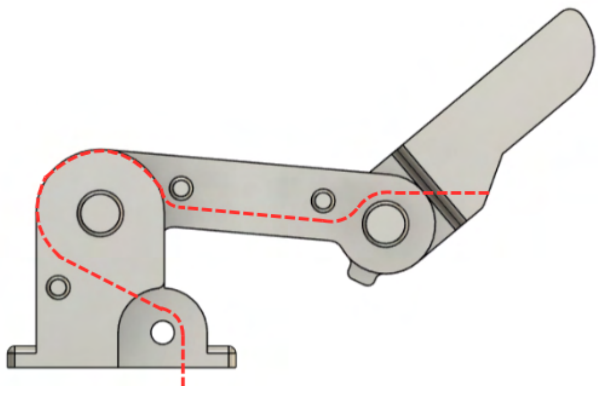


Figure 6: An overview of the finger and the tendon routing, rendered from the design in Autodesk Fusion. The red dashed line shows the tendon, controlling both of the pivotal joints in the fingers.

This finger design allows for a single tendon routed through each finger to control both finger joints, enabling an automatic adaption to the object shape when grasping. This, together with the design of the rest of the pulley-tendon system, allows for adaptive grasping with relatively high dexterity with simple control driven by a single servo.

At the distal surfaces of the fingers, thin film pressure sensors are mounted for sensitive feedback on grip pressure while grasping objects. The sensors used are SEN0294 thin film pressure sensors, with a pressure measuring range of 20 g - 6 kg. In Figure 7, the electrical schematic used to connect the force sensing resistors (FSRs) is shown.

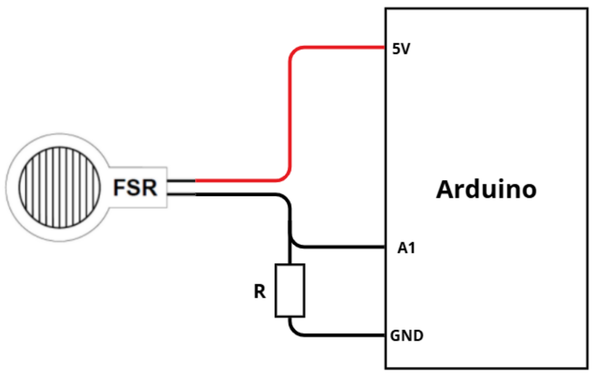
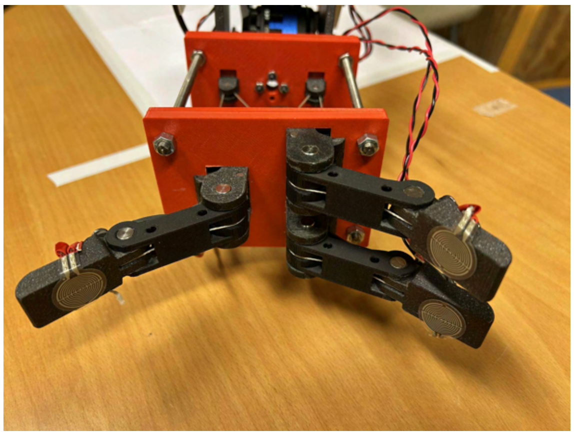


Figure 7: Electrical schematic of the FSRs connection to the Arduino. The pull-down resistance R was set to $22k\Omega$.

The circuit is a simple voltage divider, which gives an output read by analog pins on the Arduino according to

$$V_{out} = \frac{R}{R_{FSR} + R} V_{in} \tag{1}$$

where V_{out} is the output voltage, V_{in} is the circuits input voltage (in this case 5V), R is the $22k\Omega$ pull-down resistor and R_{FSR} is the FSRs resistance. The value of $R = 22k\Omega$ was set somewhat arbitrarily, and larger resistance values lead to better precision for lower pressure ranges and vice-versa [4]. Since FSRs exhibit nonlinear output characteristics, the specific cut-off, or highest allowed values, were set manually during testing based on observed sensor saturation points. A view of the fingers in the final assembly with attached FSRs, the associated electrical wiring and the rubber bands used for finger extensions can be seen in Figure 8. The FSRs were attached with their adhesive sides to the fingers surfaces, and the electrical wires were routed through the rubber bands on the backside of the fingers in order to maintain a stable position.


(a) A front view of the fingers in the final assemble.

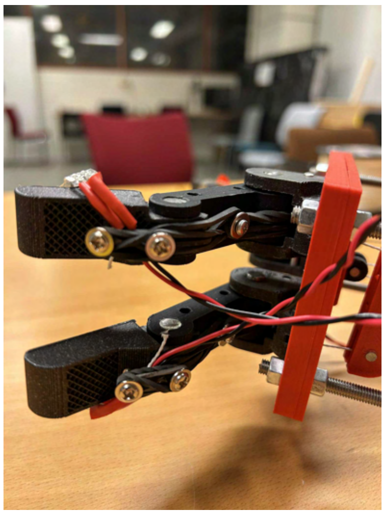

(b) A backside view of the fingers in the final assemble.

Figure 8: The fingers in the final assembly, showing the placement and wiring of FSRs and the attached rubber bands for finger extension.

3 Theory

In this section, the theory used for the object localization and object detection algorithms is presented.

3.1 Pinhole Camera Model

The pinhole camera model is a fundamental concept in computer vision, providing a geometric framework for understanding image formation. It describes how a 3D point in space is projected onto a 2D image plane through a single point aperture, assuming no lens distortion. Mathematically, the model relates the coordinates of a point in the camera frame (X, Y, Z) to its image coordinates (u, v) via the camera's intrinsic parameters: focal lengths f_x, f_y and principal point (c_x, c_y). The projection equations are given by:

$$u = f_x \frac{X}{Z} + c_x, v = f_y \frac{Y}{Z} + c_y.$$

Conversely, given the pixel coordinates and the depth Z, the corresponding 3D coordinates in the camera frame can be recovered [5].

3.2 Camera Calibration Using a Checkerboard Pattern

Camera calibration is a process in computer vision that estimates the intrinsic parameters of a camera, enabling accurate 3D scene reconstruction and image rectification. One widely used method for calibration employs a planar checkerboard pattern, a technique introduced by Zhang [6]. This approach utilizes a checkerboard with a known grid of squares, where the intersections of the black-and-white squares serve as precise control points. The camera matrix (intrinsic parameters) and distortion coefficients are calculated by minimizing the re-projection error between the observed 2D points and the projected 3D points.

3.3 Object Detection

A model for detecting objects based on text prompts is the YOLOE-11L model. It uses the YOLOv11 base detector: a residual CNN backbone for features, a PAN-FPN neck for multi-scale fusion, and a decoupled, anchor-free head. The PAN-FPN neck merges feature maps from different layers, combining detailed and abstract information. The decoupled head separates classification and localization. YOLOE adds three modules for open-vocabulary: *RepRTA* for text prompts, *SAVPE* for visual prompts, and *LRPC* for prompt-free mode. We use text prompts. RepRTA aligns pre-trained text embeddings with YOLO visual features using a small auxiliary MLP. It is re-parameterized, meaning it is simplified and merged into the main network at inference, so there is no extra speed cost. Text embeddings come from CLIP, a vision–language model that maps images and text into the same embedding space. This means the object you seek does not have to be in the labeled training set [7].

4 Method

This section introduces the materials, methods and algorithms used to achieve the project's stated goal of creating an autonomous robotic arm capable of flexible grasping, real-time sensor feedback, and precise adaptive manipulation.

4.1 Bill of Materials

3D-Printed Components

Table 1: 3D-printed components fabricated in PLA. Filament was sourced free of charge from a team members own inventory and from the 3DTeamet student lab.

Component	**Qty**	**Material / Supplier**	**Notes**	**Cost (SEK)**
Links (arm segments)	4	PLA filament, Bambu Lab A1 / Team member	20 × 5 × 0.5 cm, structural segments	0
Camera mount U-bracket	1	PLA filament, Bambu Lab A1 / Team member	Custom bracket for camera support	0
Counterweight string guides	2	PLA filament, Bambu Lab A1 / Team member	Tensioning cable guides for spring system	0
Pulley system housing	1 set	PLA filament, Prusa MK3.5S+ / 3DTeamet student lab	Dual-plate tendon housing	0
Finger linkage sets	3 sets	PLA filament, Prusa MK3.5S+ / 3DTeamet student lab	Two-joint adaptive finger design	0
Total				**0**

Commercial Structural Components

Table 2: Commercial structural components with supplier information and cost.

Component	**Qty**	**Supplier / Reference**	**Notes**	**Cost (SEK)**
Servo mounting rack	4	Course lab	Standard metal mounting brackets	0
U-bracket	5	Course lab	Steel, U-shaped servo connectors	0
Threaded rod	4	3DTeamet student lab	M5, 120 mm length	0
Stainless steel axle, 1/4" × 1"	3	Electrokit, Art.nr 41013802	Used directly as 16 mm pivot shafts for finger joints	36
Stainless steel axle, 1/4" × 12"	1	Electrokit, Art.nr 41013810	Cut into three 16 mm shafts for finger joints	69
Stainless steel axle, 1/8" × 12"	1	Electrokit, Art.nr 41012307	Cut into 15 shafts: three 25 mm and three 16 mm for finger joints, and nine 10 mm for pulley-tendon components	28
Rubber bands	9	Team member	Tensioning elements for finger extension	0
Nylon string (fishing line)	1 roll	Amazon: Dingbear 105 lb braided line, 0.5 mm × 400 m	Main tendon for pulley system	250
Total				**383**

Electronics

Table 3: Electronic components with supplier references and cost.

Component	**Qty**	**Supplier / Reference**	**Specifications**	**Cost (SEK)**
Arduino Uno	1	Course lab	Microcontroller for servo communication	0
Servo Motor - Arm joints	3	INJORA INJS035-270 Digital Servo	35 kg·cm torque, 270°rotation	735
Servo Motor - Base	1	Hitec HS-422, reused from Hubert	4.1 kg·cm torque, 180°rotation	0
Continuous Rotation Servo - Gripper	1	INJORA INJS035-360 Digital Servo	35 kg·cm torque, 360°continuous rotation	261
Force Sensing Resistors (FSR)	3	DFRobot SEN0294, Mouser Electronics	20 g - 6 kg range	146
Resistors ($22k\Omega$)	3	3DTeamet student lab	Voltage divider pull-down resistors	0
Webcam	1	Logitech C270, course lab	1280 × 720 USB camera	0
Total				**1 142**

The total cost of materials for the project was 1525. This section refrains from mentioning basic assembly components such as screws, bolts and nuts, but all of these components where sourced free of charge from the course lab and the 3DTeamet student lab.

4.2 Hardware Manufacturing and Assembly

The mechanical components of the robotic arm were manufactured primarily using 3D printing and off-the-shelf mechanical parts available through the course lab or other student labs. All CAD models were designed using Autodesk Fusion, allowing for a collaborative design process and consistent geometry between different parts.

3D printed parts include the arm links, pulley-tendon system components, finger phalanges and the U-bracket for camera mounting. These parts were printed in PLA filament using a Bambu Lab A1, Prusa CORE One and a Prusa MK3.5S+ printer available through team members (Bambu Lab printer) and the 3DTeamet student lab at Chalmers (Prusa printers). The slicing was done using the PrusaSlicer and Orca Slicer software, with varying print settings for different parts.

Post-processing of 3D printed parts involved light sanding, filing and hole reaming to ensure precise fitment between parts. Additionally, the threaded rod and pivot linkages were cut to size using a Dremel multipurpose tool. All of the manufacturing and assembly process was carried out in the course lab and the 3DTeamet student lab, using the available tools in the lab.

The mechanical assembly was done in several steps: (1) assembly of the arm structure with associated servos and electrical wires, (2) mounting the fingers and the pulley tendon system, (3) mounting the hand to the arm and (4) setting the FSRs in place and routing their electrical wires. The assembly process was however carried out in non-linear order, since certain parts had to be replaced or redesigned.

4.3 Firmware Implementation and Calibration

The Arduino firmware was implemented to receive pulse width modulation (PWM) signals from a computer through a USB serial connection. The Arduino sent back a boolean indicating whether the joint had reached its desired position or not. The firmware moved the servos incrementally toward the target angle and supported moving several joints simultaneously.

A calibration was performed to relate the PWM of the servos to their angles by measuring angles at specific PWM values or finding the PWM values for easily measured angles such as 0°, 45°, and 90°. The relationship was found to be linear. This mapping was done on a separate computer so that the algorithms could operate in angles while the Arduino only received PWM values.

4.4 Manual Control

For further testing and calibration, a Sony DualSense (PS5) bluetooth controller was used to manually operate the arm via a Python interface. Using the *pygame* library, the script translates joystick and button signals into PWM commands that are sent over serial to the Arduino, which then drives the servos accordingly. Each joystick axis and trigger is mapped to specific joints of the arm (e.g. base, shoulder, elbow, wrist, and gripper). Joystick/trigger deflection determines both the direction and speed of the corresponding servo, allowing for precise and independent control of each joint. Additional buttons provide safety functions such as freezing all movements and moving back to a safe neutral position.

During operation, the script continuously also monitors the pressure sensor data received from the three mounted thin-film pressure sensors on the tip of the fingers. If excessive pressure is detected on any of the three, an automatic fail-safe halts the movement of the fingers. Depending on the object to be grasped, this threshold could then be adjusted.

4.5 Object Detection

For object detection, several approaches were evaluated. Simpler methods were used first, followed by more advanced and accurate ones. This allowed early testing of the integration between object detection and other parts, such as object localization, and enabled comparison of the different methods and their efficiency.

The first approach used OpenCV to detect edges with *cv2.canny* and then searched for connected edges, or contours, forming a surface. If the surface was large enough, it was considered an object. This simple method worked under the assumption that the background was a table or similar flat surface.

The method used in the final setup, see Section 4.9, was YOLOE-11L, see Section 3.3. Another method considered was using the YOLOv11 model or similar for objects already included in its training set. Since not all objects considered in this project were part of this set, fine-tuning a model such as YOLOv11 for each object was also explored. However, using YOLOE-11L was a faster and easier option since it is open-vocabulary.

An object was considered detected if the model's confidence was above 0.2, where 0 is the minimum and 1 the maximum confidence.

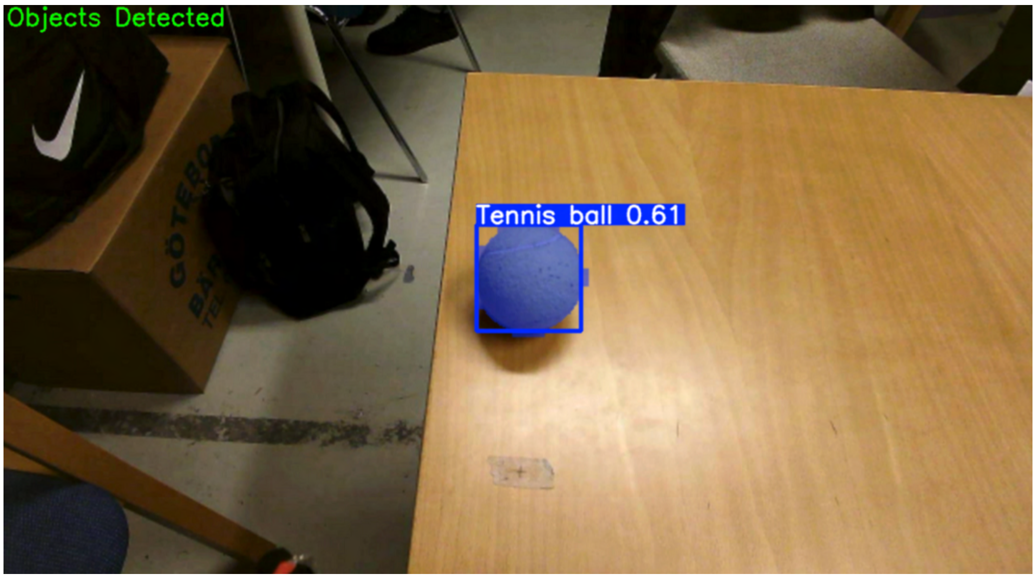


Figure 9: Example of YOLOE-11L detecting an object and calculating a bounding box around the object, here shown in blue. The number to the right of the text prompt "Tennis ball" is the confidence the model is showing.

4.6 Object Localization

Object localization was carried out using a monocular camera with a resolution of 1280×720 pixels mounted on the robots end-effector. The use of a single camera inherently limits direct depth estimation, necessitating that object positions be inferred from geometric assumptions rather than explicit 3D measurements. By assuming that all free objects lie on a planar surface at a known height, approximate positions can be computed.

Using the pinhole camera model described in Section 3.1, a direction vector $\mathbf{d}_{camera}$ from the camera frame to an object can be derived from its pixel coordinates u and v.

$$\mathbf{d}_{camera} = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{u - c_x}{f_x} \\ \frac{v - c_y}{f_y} \\ 1 \\ 1 \end{bmatrix}$$

To improve the accuracy of this direction vector, the pixel coordinates and camera intrinsics were undistorted using the camera calibration procedure outlined in Section 3.2 (implemented with OpenCV) to compensate for the lens distortion.

The direction vector is transformed to the world frame by:

$$\mathbf{d}_{world} = T_{cw}\,\mathbf{d}_{camera} - \mathbf{b} = T_{cw}\,\mathbf{d}_{camera} - T_{cw}\begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

where T_{cw} is a homogeneous transformation matrix that converts points from the camera frame to the world frame, given the robot joint angles. The term $\mathbf{b}$ describes the position of the camera frame in world coordinates.

$\mathbf{d}_{world}$ defines the direction of the line:

$$\mathbf{l}(t) = t\mathbf{d}_{world} + \mathbf{b}$$

The object is assumed to lie on a plane at height $z = z_0$ in the world frame. The intersection parameter t is obtained from the z-component of the line:

$$t\mathbf{d}_z + \mathbf{b}_z = z_0 \quad \Rightarrow \quad t = \frac{z_0 - \mathbf{b}_z}{\mathbf{d}_z}$$

Substituting back into the line equation gives the position of the intersection point:

$$\mathbf{p} = \mathbf{l}\left(\frac{z_0 - \mathbf{b}_z}{\mathbf{d}_z}\right) = \mathbf{b} + \frac{z_0 - \mathbf{b}_z}{\mathbf{d}_z}\,\mathbf{d}_{world}$$

To obtain a physically valid intersection with the plane, the direction vector must point toward the plane in the world frame. For example, if the ray originates above the plane ($\mathbf{b}_z > z_0$), then it must satisfy

$$\mathbf{d}_z < 0.$$

4.7 Inverse Kinematics

To put the end-effector at a desired position and orientation, the arm joint angles are determined using a feed-forward neural network. The structure of the network is a simple multi-layer perceptron with an input dimension size of three (Cartesian position coordinates), three hidden layers of size 156 and an output dimension size of four (number of arm joints). All layers except the output use the ReLU activation function. Since the angles of each arm joint are constrained, each output x_i is scaled to its allowable range according to the following activation function:

$$f_i(x_i) = \left(\frac{\tanh(x_i) + 1}{2}\right)\left(\max_i - \min_i\right) + \min_i, \quad i = 0, 1, 2, 3$$

where min and max outline the allowed range for the specific joint i.

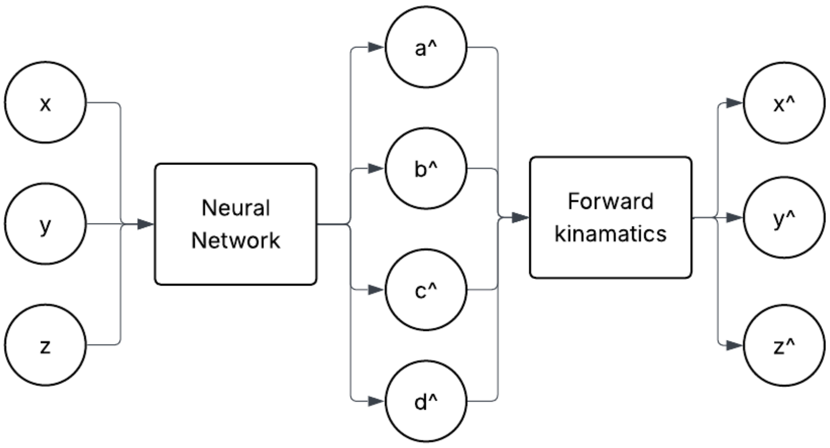


Figure 10: **Overview of the Inverse kinematics architecture.** x, y and z represents the cartesian end-effector position. $\hat{a}$, $\hat{b}$, $\hat{c}$ and $\hat{d}$ represents the predicted joint angles. $\hat{x}$, $\hat{y}$ and $\hat{z}$ are the reconstructed cartesian end-effector position given by the predicted joint angles.

The network is trained analogously to an autoencoder, where the forward kinematics function serves as a fixed decoder that maps the predicted joint angles back to a reconstructed end-effector position, see Figure 10. In addition, the generated joint angles must result in an end-effector orientation suitable for grasping objects effectively. The desired orientation was defined as the end-effector (the robotic hand) being parallel to the z-axis, enabling objects to be picked up from above. To induce this behavior in the network, the loss function was constructed as follows: We consider the following reference points in the local hand frame:

$$\mathbf{p}_0 = \begin{bmatrix}0\\0\\0\end{bmatrix}, \quad \mathbf{p}_1 = \begin{bmatrix}1\\0\\0\end{bmatrix},$$

where $\mathbf{p}_0$ is the origin of the hand frame and $\mathbf{p}_1$ defines the forward direction of the hand.

Let $T_{\text{hw}}(\mathbf{p}, \hat{\boldsymbol{\theta}})$ denote the transformation of a point $\mathbf{p}$ from the local hand frame to the world frame, given the predicted joint angles $\hat{\boldsymbol{\theta}}$. Using this, the hand's forward direction $\mathbf{d}_{\text{forward}}$ in world coordinates is expressed as:

$$\mathbf{d}_{\text{forward}} = T_{\text{hw}}(\mathbf{p}_1, \hat{\boldsymbol{\theta}}) - T_{\text{hw}}(\mathbf{p}_0, \hat{\boldsymbol{\theta}}),$$

and its z-component is expressed as:

$$\mathbf{z} = (\mathbf{d}_{\text{forward}})_z.$$

Let $\hat{\mathbf{p}}_{\text{EE}}$ be the predicted end-effector position and $\mathbf{p}_{\text{EE}}$ the corresponding target position. The complete loss function is defined as:

$$\mathcal{L} = \mathbb{E}\Big[\|\hat{\mathbf{p}}_{\text{EE}} - \mathbf{p}_{\text{EE}}\|_2^2 + \lambda\,(\mathbf{z}+1)^2\Big], \quad \lambda = 0.005,$$

where the first term encourages accurate end-effector positioning, and the second term is a regularization encouraging the hand to point downward ($z \approx -1$).

The training dataset of end-effector positions was generated by sampling random joint angles within their specified limits and subsequently filtering the samples based on the hand direction vector, such that the z-component satisfies $z + 1 < 0.01$ ($\approx$ 8 degrees from downward z-axis).

4.8 Movement and Grasping

When an object was localized the arm moved to a locations slightly above the object to not knock it over. It then descended slowly until it was at a hardcoded distance above the object. Then a set PWM of 1350 μs was sent to start the gripping process. After 12 seconds the PWM 1500 μs was sent to servo making it stop and hold the position. The object was then lifted straight up, to not drag the object on the table, and then moved to a set end position. At this point the PWM 1650 μs was sent to the servo making it release the object. After 7 seconds the PWM 1500 μs was sent to the servo again making it stop. The time of releasing being shorter than the grasping was a result of that when grasping the string barely shortens anything after that the hand starts touching the object, but it needs to keep griping for a while to get a better grip.

4.9 Test Setup

To evaluate the arm's performance, the setup in Figure 11 was used. It included the arm, a computer running the algorithms, and a paper showing the arm's workspace on a horizontal plane at a height of 7.5 cm. The object, which was a tennis ball placed on a white roll of tape to keep it still, was positioned at this height. The prompt sent to the YOLOE model was "yellow-green ball," since this gave higher confidence than "tennis ball", see Figure 9 for example. For further parameters used, see *New_Arm/main.py* in the GitLab repository found here in Section 8.

The arm rotated it's base servo position until the object came into frame, it then proceeded according to Section 4.8.

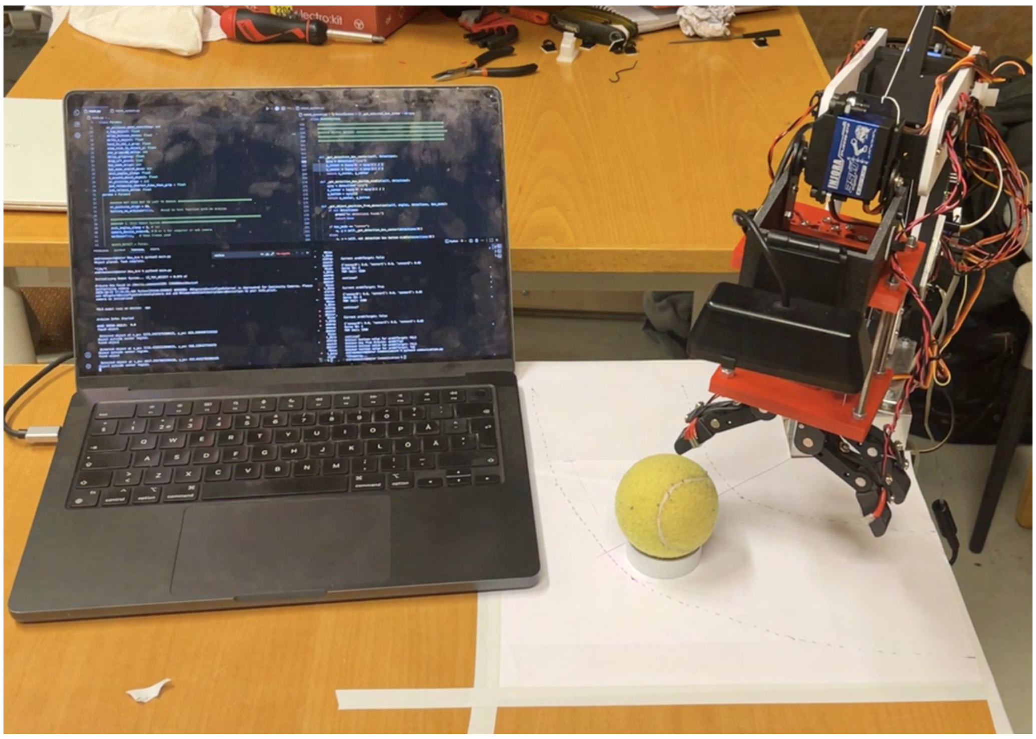


Figure 11: The setup used for testing the arm's capabilities.

5 Result

The arm supported control with a PS5 controller, automatic point to point motion, text to object detection, depth estimation with a pinhole camera model requiring a fixed object plane and height, and force sensing with a set threshold. In Figure 12, a collage of the arm operating in both automatic and teleoperated mode is shown. See Appendix for full video demonstration.

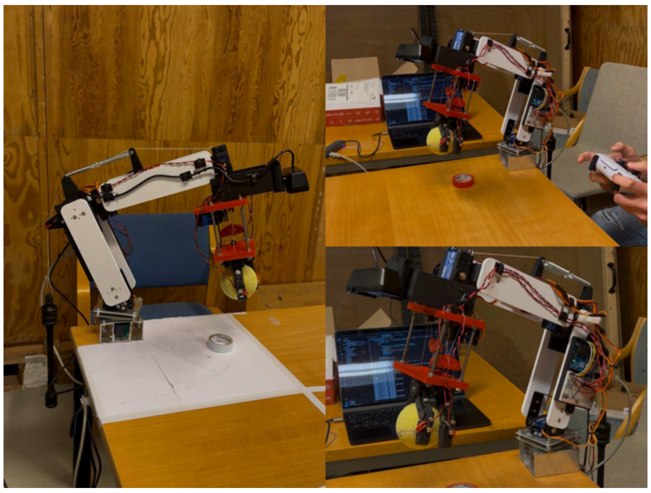


Figure 12: Collage of arm under operation.

Servo precision was limited, especially at the base, likely due to overload, which also reduced the usable workspace, see Figure 13 and 14. One design flaw was that the camera position caused targets to leave the frame during grasping, which made accurate targeting more challenging. The force sensors were also not integrated in the autonomous control, which reduced the arm's capability to safely handle fragile objects as originally planned.

5.1 Workspace

The workspace of the robotic arm can be seen in Figure 13, illustrating the inner and outer workspace radii of the robotic arm at different end-effector heights, and in Figure 14 showing a view of the horizontal workspace at a height of 7.5 cm.

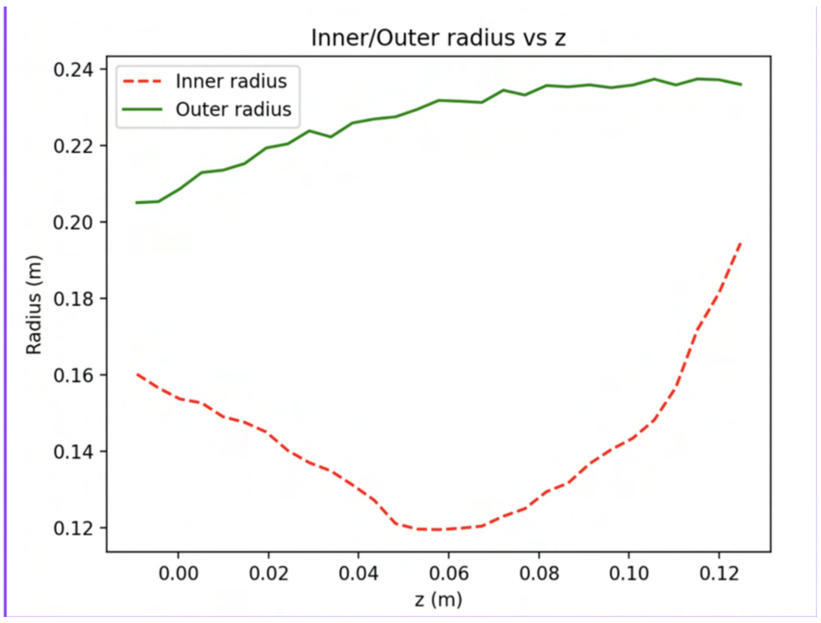


Figure 13: Inner and outer radius for the workspace as a function of the height above the table z.

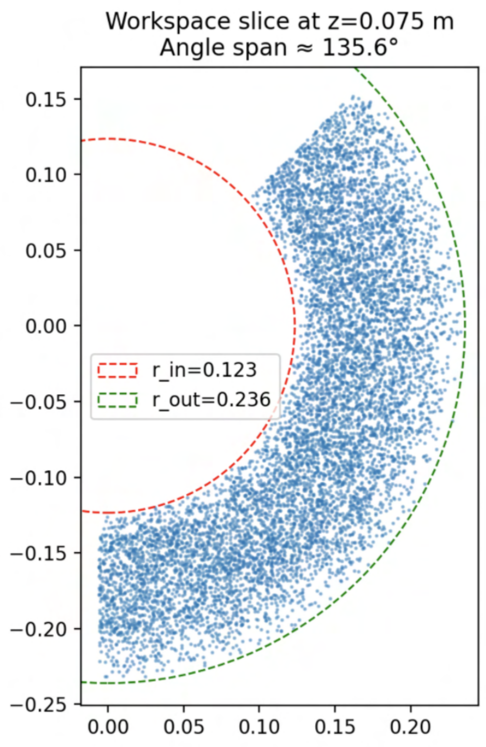
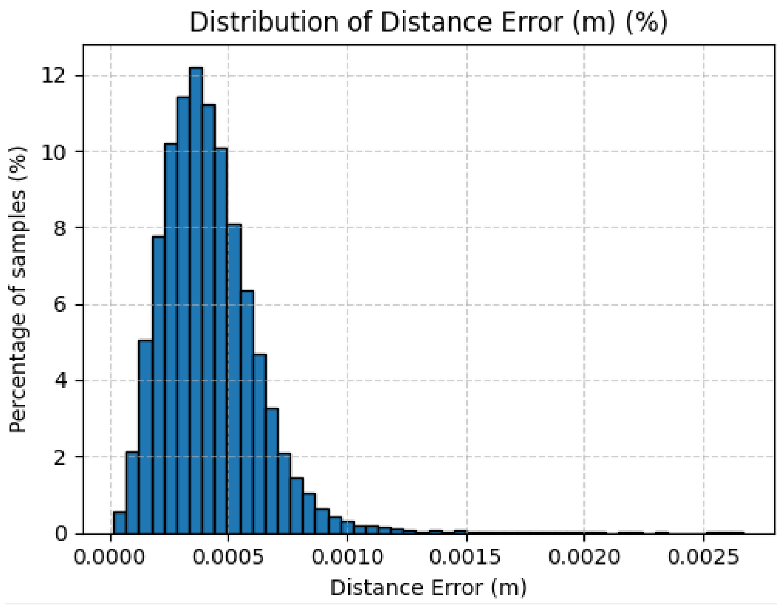


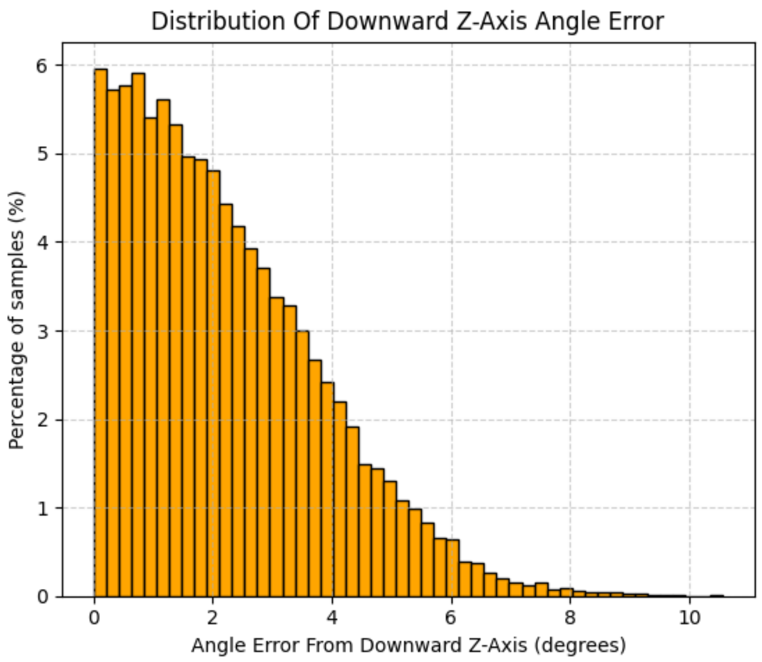
Figure 14: Horizontal plane at 7.5 cm above the table. Blue dots are showing some sampled workspace points that were used to calculate the radii limits. The axes are in meters.

5.2 Results of the Trained Inverse Kinematics Model

In Figure 15, the distribution for reconstructed end-effector positions from predicted joint angles (a), together with the dstribution of end-effector direction angles relative to the downward z-axis as derived from the predicted joint angles (b), are shown.



(a) Distribution of Euclidean distance errors for reconstructed end-effector positions from predicted joint angles.



(b) Distribution of end-effector direction angles relative to the downward z-axis (world frame) as derived from the predicted joint angles.

Figure 15: Histograms for Trained Inverse Kinematics Model. Generated from a sample of 50,000 end-effector (EE) positions obtained from random joint angles, where the z-component of the EE direction vector (world frame) satisfies $z + 1 < 0.01$ ($\approx$ 8 degrees from downward z-axis).

6 Discussion

The following section discusses the overall performance and limitations of the developed robotic arm system, encompassing its inverse kinematics model, object localization method, hardware design, and integrated system behavior.

6.1 Inverse Kinematics Model Performance

The performance of the neural network-based inverse kinematics model demonstrates strong accuracy overall. As illustrated in Figure 15 (a), the vast majority of the distribution of Euclidean distances between the original and reconstructed end-effector (EE) positions lies below 1 mm. Only a small number of outliers exceed this threshold, with the largest observed deviation being approximately 2.5 mm. This indicates that the model is capable of reconstructing EE positions with sub-millimeter precision in most cases. In comparison, the neural network model seems to outperform the numerical Jacobian transpose method, which was explored as an alternative inverse kinematics solution. The Jacobian-based approach struggled to achieve positional accuracy better than 1 cm, with some outliers reaching deviations of up to 5 cm. In addition to accuracy, the neural network method offers the advantage of consistent computation times. The model also manged to reasonably capture the end-effector orientation behavior of pointing toward the object plane, in order to enable reliable grasping. As shown in Figure 15 (b), the majority of reconstructed positions had an orientation angle of less than 8° relative to the downward z-axis.

6.2 Object Localization Capabilities and Limitations

The current object localization system demonstrates the ability to identify object positions with an accuracy on the order of a few centimeters relative to their geometric centers. Nevertheless, it is subject to significant variability, influenced by several factors. The main fundamental factor influencing accuracy is the observation angle. When the angle of incidence is high and a pixel near the center of the object is selected, the estimated position in the x-y plane tends to be biased behind the object's actual position. This effect can be partially mitigated by selecting a pixel closer to the bottom of the object; however, in this case, the achievable precision is inherently limited to approximately half the geometric size of the object. Another potential mitigation strategy, applicable when the object height is known, is to raise the virtual floor plane, thereby reducing the positional bias caused by high observation angles. Additional sources of error that influence positional accuracy are uncertainties in the joint angles and inaccuracies in the camera's intrinsic parameters. Among these factors, uncertainties in the joint angles have been the greatest challenge. The design choice of mounting the camera on the end-effector was motivated by the ability to iteratively navigate toward the object and to observe it from multiple perspectives, thereby improving position estimation and mitigating some of the issues described above. However, these methods were not fully developed in the current system, partly due to the limited range of the robotic arm.

6.3 Hardware Design

The hardware platform of the robotic arm designed and assembled in this project provides the intended function, capable of fully moving within it's workspace and grabbing different shaped objects. The decision to use standardized metal servo brackets 3D printed PLA components provided a balance between sufficient structural rigidity and the ability of rapid, iterative development. However, the joints could benefit from being more rigid, since the current assembly has some sway even when the servos are locked in position, making the control and movement of the arm less precise.

The counterweight and spring-assisted elbow system proved effective in reducing the load on the shoulder and elbow servos, though it took several iterations to fine-tune the stiffness of springs and mass of the counterweight to get sufficient function. Despite the improvement, the servos still operated close to their limits when lifting objects or moving the arm in extended positions. This limits the workspace and the arms capability of lifting heavier objects, and lead to occasional positional drifts and thermal overload of the servos. Future improvements could include using more rigid, custom-made joint constructions and perhaps planetary gears or similar systems improving the torque of the joints. An alternative to planetary gears could be to simply use stronger servos, and positional encoders could be an option to eliminate the potential problem with positional drift and further improve the preciseness of control.

The end-effector hand performed as expected, enabling adaptive grasping through passive load balancing between fingers. It allowed for easy control being an underactuated system driven by only one servo, and drawing inspiration and building upon the Yale OpenHand Project allowed for quick development. However, mechanical friction in the tendon routing, caused by a lack of nylon pulleys or similar pulley parts, and finger joints friction slightly limited smoothness and symmetry of motion. Additionally, the servo used being a 360° continuous rotation servo without any positional feedback limits the ability of control to using time-based flexion and extension of the fingers, leading to positional drift over several gripping cycles and lack of control of the fingers positioning when grasping. In the current design, a non-continuous rotation servo could be used, with positional control allowing for more precise control. The control of grasping could also be improved by using a positional encoder, or using a more high-end servo with positional feedback.

Overall, the hardware design successfully validated the core mechanical concepts and achieved the sought functionality. It serves as a functioning robotic arm platform for vision-based control and neural inverse kinematics, and the combination of simplicity, modularity and adaptability makes the platform a strong foundation for continued development towards a more capable humanoid robotic arm. Many limitations of the current design stem from compromises needed in order to achieve a rapid enough prototyping to obtain a functioning arm within the course of time for this project.

6.4 Overall System Performance

The multi-purpose mechanical arm can automatically locate and move objects within its workspace, see Figure 13 and 14. It can also be controlled manually with a PS5 controller. However, the lack of integration of feedback from the force sensors in the automatic control means the gripping must be hardcoded for each specific object. As a result, the arm cannot handle objects of different fragility automatically. Another effect of missing force feedback is that the height of each object must be roughly known in advance. With force sensor feedback, the arm could instead start above the object and move down slowly until a suitable force reading is detected. However, given that force sensor feedback was successfully integrated in the manual control of the arm with the PS5 controller, the functionality itself is proven and could with relatively low effort be integrated in the automatic control software.

The arm can move fairly fast when using the setup in Figure 11. However, in the demonstration video, its speed is limited by delays implemented in the movement program. These delays could be reduced significantly, but that would require further parameter optimization. It is estimated to be able to perform tasks at about a third of human speed. To improve this, the software modeling of the arm needs higher accuracy, and the servos need to be stronger to hold their positions better. The gripper might also require suction, adhesive, or a completely different hand topology to handle fragile objects quickly without dropping them.

Despite these limitations, the current arm is useful for repetitive tasks involving predetermined objects. However, durability may be an issue. The elbow joint is a weak point, it overheats quickly due to overload, which could limit long-term use. Another issue is that the string tension in the hand may drift after several grip-and-release cycles. This happens because the 360° servo operates by velocity rather than position. A simple fix could be to hardcode a specific release time for each object, or just a reference object, so that the tension of the string is known before releasing.

7 Conclusion

A multi-purpose arm based on the Hubert platform was developed. It is able to automatically locate, grasp, and move objects using underactuated adaptive fingers. Force sensor feedback was not implemented in the control program for automatic object handling, but proved it's functionality in manual control using the PS5 controller. The current arm can perform certain repetitive tasks, but its durability and longevity is somewhat limited. Overall, the project demonstrated that a low-cost robotic arm can achieve a meaningful level of dexterity and autonomy using open-source tools and off-the-shelf components. The integration of computer vision and neural network based inverse kinematics validated the concept of intelligent control in an affordable hardware platform. While limitations in strength and sensing remain, the system serves as an effective proof of concept for modular and rapid development, with good potential for further improvement.

8 Future work

The features of this project were limited by time and money but it doesn't mean that it doesn't have evolution perspective or future work that could have been made without these two restrictions.

First, stronger servos can be an important investment to make. It would allow a better design without springs or counter weights. It would also allow the arm to have a better reach by using longer links, and increasing the ability to grab heavier objects. Another potential improvement within this area is to use more robust joints with some method of increasing torque, for instance using planetary gears or similar concepts.

The design of the end-effector hand of the robot also has potential areas for future improvement. At the price of a few CAD modifications, an improved design with more fingers, but still using only one servo, could be created. This would permit a better grasp and thereby less risks of letting go of an object. More segmented fingers drawing further inspiration from human fingers could adapt better to more complex object shapes, but it would potentially require stronger and more precise servos with positional feedback. Additional material such as nylon pulleys or bearings could also reduce friction and facilitate smoother movements. The hand could also be improved by updating the design for minimal weight, reducing the load of the servos and increasing the arm's overall capability of movement and object manipulation.

Further improvements in robot control and behavior could be achieved by integrating additional sensory modalities, such as proprioception. Access to precise joint angle data obtained from direct measurements would enable more accurate estimation of the robot's pose and configuration. This, in turn, would lead to better object localization and manipulation accuracy, as these processes depend directly on kinematic transformations. Furthermore, richer perceptual input could enable the development of more sophisticated control algorithms and behaviors. For example, it could support the construction of internal world models that capture environmental dynamics and the consequences of the robot's actions. These models could act as simulated environments, providing a foundation for reinforcement learning approaches that optimize goal-directed behavior requiring less costly and time-consuming real-world interactions [8] [9].

Appendix

Code and Video Demonstration

Link to code and video on GitLab.

References

[1] Universal Robots. Ati axia80 force torque sensor package, 2025. Accessed: 2025-10-08.

[2] Venkatesh Pattabiraman, Zizhou Huang, Daniele Panozzo, Denis Zorin, Lerrel Pinto, and Raunaq Bhirangi. eflesh: Highly customizable magnetic touch sensing using cut-cell microstructures. https://e-flesh.com/, 2025. Accessed: 2025-10-08.

[3] Yale University, Grablab / Yale OpenHand. Yale openhand project — model t. https://www.eng.yale.edu/grablab/openhand/model_t.html, 2025. Accessed: 2025-10-08.

[4] Jinwoo Jung, Kihak Lee, and Bonghwan Kim. Precision calibration and linearity assessment of thin film force-sensing resistors. *Applied Sciences*, 14(16), 2024.

[5] Richard Hartley and Andrew Zisserman. *Camera Models*, page 153–177. Cambridge University Press, 2004.

[6] Zhengyou Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22:1330–1334, December 2000. MSR-TR-98-71, Updated March 25, 1999.

[7] Ultralytics. Yoloe model — architecture overview. https://docs.ultralytics.com/models/yoloe/#architecture-overview, 2025. Accessed: 2025-10-26.

[8] David Ha and Jürgen Schmidhuber. World models. https://doi.org/10.48550/arXiv.1803.10122, 2018.

[9] Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models. https://arxiv.org/abs/2301.04104, 2024.