Integration and development of AI Accelerator applications in Universal Robots arm.

Xavier Soto Picón

June 30, 2025

Abstract

This work presents an autonomous robotic system to enhance the autonomy of individuals with reduced mobility through voice-controlled object manipulation. The architecture integrates a pipeline of AI models on a Universal Robots arm, all within its new NVIDIA Jetson AI Accelerator. It processes voice commands using Whisper and a Small Language Model, detects objects with a fine-tuned YOLOv8, and computes the ideal grasp pose using SAM segmentation masks achieving a 4 second end-to-end latency with high accuracy. This enables reliable picking of a variety of objects with different shape and sizes. Overall, the project serves as a successful proof of the immense potential that arise when combining AI and robotics for high-performance assistive applications. The complete source code is available at https://github.com/xsotopi/TFG.

Keywords: Assistive Robotics, Voice Control, Deep Learning, AI, Computer Vision, YOLOv8, SAM, Small Language Model, Hand-Eye Calibration, Camera Intrinsics, Human-Robot Interaction, Real-Time Systems, AI Accelerator.

1. INTRODUCTION

Artificial intelligence (AI) has been the latest technological boom, with Large Language Models (LLMs) like ChatGPT or autonomous driving systems among many other advances that have revolutionized humanity [1]. However, before its rise, one of the main focuses was robotics, with also an incalculable potential on what can be achieved and the infinite different way that it can help society grow. In this project, I will combine these two fields to develop an innovative application that uses the strengths of AI and robotics [2], addressing real-world challenges and contributing to a more inclusive and efficient society.

1.1. PROJECT DESCRIPTION

This work focuses on designing and developing a robotic assistance system able to help and enhance the autonomy of people with some disabilities or reduced mobility. The system allows the user to control a Universal Robots collaborative robotic arm through natural voice interaction using Whisper, an OpenAI model, to transcript the voice, and Qwen3 to extract commands of pick and place within the audio prompts. To detect and classify objects YOLOv8 is used.

By facilitating object handling without requiring a direct physical interaction, this system can significantly improve the quality of life for individuals with physical disabilities, such as motor impairments, enabling them to perform everyday tasks more independently.

This technology could be implemented in homes to support the users in daily activities such as picking up objects, organizing spaces, or even assisting with feeding. Additionally, it could be also used in care centres, where it would help caregivers by providing a more autonomous solution when assisting residents with mobility challenges. Beyond these environments, the system could also be integrated into workplaces, rehabilitation centres, public facilities or supermarkets, further extending its accessibility and usability to a wider range of individuals in need.

2. OBJECTIVES

The primary objective of this project is to develop an application combining both AI and Robotics that works autonomously, controlling it by voice commands, without requiring pre-programmed movement patterns for the robot, detecting the different objects in an image, recognizing them, picking and placing the object to a target location.

- Contact E-mail: xaviminisoto@gmail.com
- Supervised by: Carlos García (Dept. Computació)
- Academic Year: 2024/25

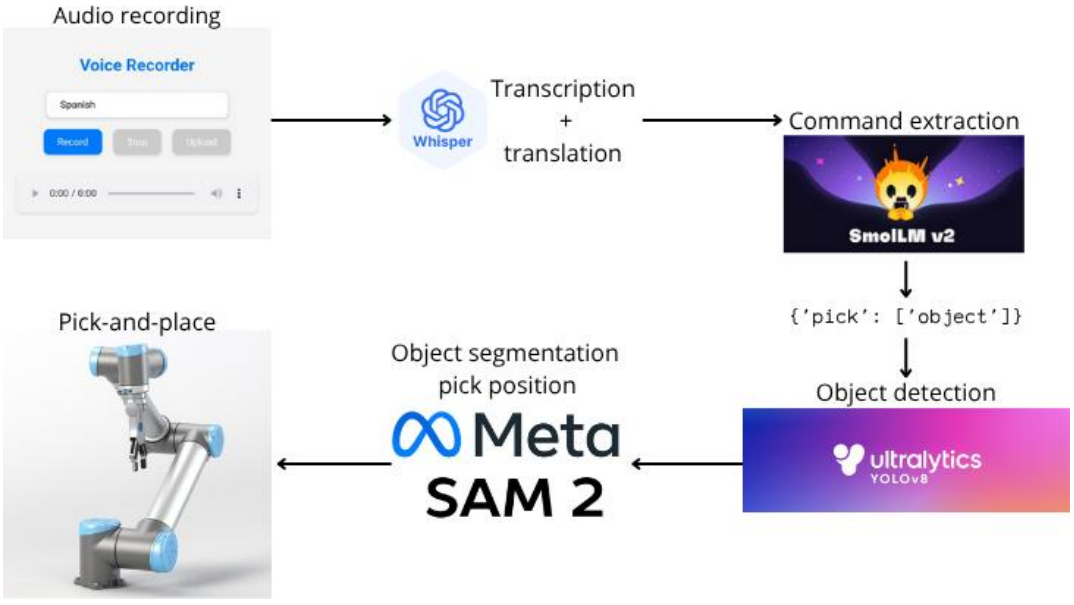

Figure 1: Pipeline of the project

To achieve this primary goal, the project is broken down into several key sub-objectives, see Appendix [A1], following a "divide and conquer" strategy, where the focus is set on smaller parts rather than on everything simultaneously addressing individually each critical module of the end-to-end pipeline.

- **Speech recognition and interpretation:** Implement a system to accurately transcribe spoken user commands into text using Whisper and then interpret and extract commands with Qwen3 small language model.

- **Object detection:** Fine-tune a YOLOv8 model capable of reliably detecting and classifying a variety of custom objects.

- **Segmentation and grasp calculation:** Use Segment Anything Model (SAM) to obtain precise segmentation masks of the object and compute an optimal grasp point and angle.

- **Robotic Integration:** stablish a robust communication pipeline between the AI system and the robotic arm, to move to computed poses.

The application has been developed by treating the robot and the external compute module as individual entities working separately and communicating with each other through sockets. The variety of AI models are computed outside the robot controller, then a pose is sent to the robot, and the robot performs the corresponding move command.

3. METHODOLOGY

The methodology followed to succeed in the development of the project is Kanban [9], an agile methodology that enables smooth task management by dividing work into smaller, manageable tasks. This method was selected due to its flexibility.

The mentioned approach has been performed in the UR Collaborative hub in Barcelona with a tutor/supervisor and all the necessary material provided.

The workflow consists of three phases:

1. **Learning:** Involves acquiring all the knowledge needed to be able to do the project. It includes both learning to control the robot and other related needed programs.

2. **Development:** The core of the project, it includes the development of the models, the robot and the connection between them.

3. **Deployment:** Consists of testing and simulating different cases in real-life scenarios as

well as finishing all the deliverables (report, presentation, code …)

With the defined objectives of the project and the methodology to be followed, a Table of Tasks found in Appendix [A2] and its subsequent GANTT in Appendix [A3] chart has been done. In the tasks can be seen the three phases with their own tasks as well as the different report deliverables are highlighted.

4. DEVICES

In this section the required devices, their role in the system, and some relevant technical specifications is explained.

4.1. ROBOT

The main hardware component of the project is the Universal Robots UR10 [11], a collaborative robotic arm designed for industrial and research applications. It enables the movement and manipulation of physical objects, allowing the pick-and-place task to be performed in the project with high precision and repeatability.

Figure 3: Robot with camera and gripper mounted

In Figure 3 can be seen the robot used with the camera attached to it.
The robot is controlled with UR own programming software Polyscope. Specifically, Polyscope X [20], their latest software environment, is used as it is required due to compatibility to the AI Accelerator. It provides an intuitive graphical interface being able to see the 3D pose of the robot, as well as make the programs without writing any code by adding movements and actions visually in the script.

Additionally, an end effector an OnRobot RG2 [15] gripper has been used. Although it is not one of the most recent models, it has proven to be perfectly effective, allowing a secure and reliable picking of a variety of objects of different shapes and sizes.

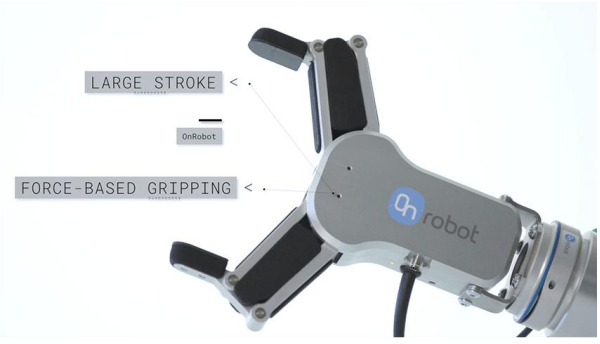

Figure 2: OnRobot RG2 gripper used

4.2. CAMERA

The camera used in the project is the Orbbec Gemini 335Lg [10], shown in Figure 4, is a 3D stereo vision camera that enable both RGB image capture and depth estimation. Additionally, it contains infrared sensors for improved depth accuracy under variable lighting.

Figure 4: Front and bottom view of the camera. It shows the optics of the camera as well as its connections

No camera comparisons have been performed, as the selected one comes with the AI Accelerator pack, already having all the requirements that a camera for reliable object detection.

This camera plays a crucial role in object detection and pose estimation. While the depth data is primarily used to compute the Z-axis (height) in robot space, the RGB is used for running the YOLOv8 object detection model.

In Figure 5 can be seen each of the different images that can be obtained from the camera, the RGB image, the infrared images, and the depth camera where object shapes can still be identified. This camera is mounted on the robot arm's end-effector flange, alongside the gripper and is secured with Velcro. Finally, it is connected with a USB 3.0 cable to the AI Accelerator to one of its SuperSpeed (SS) ports.

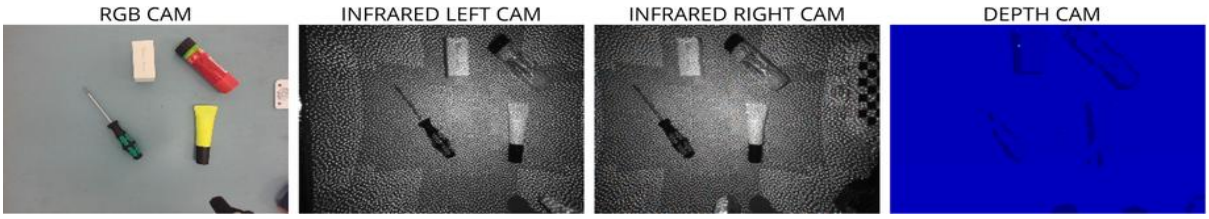

Figure 5: Different camera frames, including an RGB image, 2 infrared ones and a depth image.

4.2.1 CAMERA CALIBRATION

After mounting the camera on the robot, a precise calibration is essential to establish a mathematical relationship, in the form of a matrix, between the camera's 2D pixel coordinates and the robot's 3D world translating what the camera "sees" into a location the robot can "reach". The calibration process to obtain the camera intrinsics, and the hand-eye calibration (see Appendix [B1] for more details), consisted of:

- **Dataset Collection:** A set of 81 image-pose pairs was collected. Images consist of a picture of a calibration pattern (8x7 chessboard with 15mm squares) and poses are the robot's tool flange position at the moment of capture, having its position (X, Y, Z) and rotation (Rx, Ry, Rz). See Figure 6 for an example and for more details.

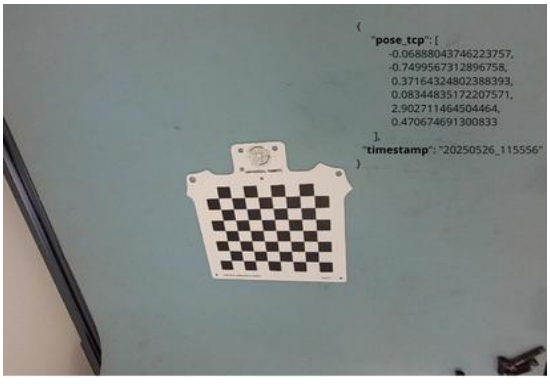

Figure 6: Example of calibration image. Can also see the exact pose of the TCP when it was taking.

- **Calibration process:** All image-pose pairs were used to perform an initial calibration. A validation script using the results then evaluated each pair to discard the "outliers" based on the reprojection error and the hand-eye consistency before recalibrating again. This process was repeated until all validation values were less than 0.5 standard deviations from the mean. See Appendix [B2] for detailed filtering used.

- **Calibration results:** With the filtering process, a final set of 34 pairs was for the final calibration. A final validation of these results showed a low translational standard deviation of 0.89mm and a high rotational standard deviation of 88º. See Appendix [B3] for a detailed explanation of the results.

This high rotational error result is acceptable for the project because the system relies on the calibration only for the highly accurate positional data. The final grasp orientation is calculated independently by analyzing the 2D object's shape while fixing the Roll and Pitch angles and ensuring the gripper is kept always parallel to the z-plane of the table. Therefore, rotational inaccuracy is irrelevant for the final pose computation.

4.3. AI ACCELERATOR

The AI Accelerator [8] is a high performance Nvidia Jetson AGX Orin 64GB compute module provided by Universal Robots. It contains a powerful integrated GPU based on the NVIDIA Ampere architecture, identified as Orin (nvgpu). This GPU has 2048 NVIDIA CUDA cores and 64 Tensor Cores, specially designed to accelerate the parallel computations required by moder AI models allowing for much faster training, and more importantly, near real-time inference for the models used in this project.

While it is not strictly required for the execution of the pipeline, its use has been fundamental due to its improvement in processing times, high compute capacity, and compatibility with the camera connection.

The primary advantage of the AI Accelerator is its powerful Graphics Processing Unit (GPU). Unlike a CPU which is designed with a few powerful cores for handling sequential tasks, a GPU contains a massively parallel architecture that consists of thousands of smaller more efficient cores. This structure makes

it ideal for the demands of AI and deep learning as their models rely on performing vast amounts of matrix and vector calculations. GPUs execute those simultaneously across their many cores drastically reducing the computation time, a critical aspect when interacting with a robotics' application that interacts with a person.

Figure 7: AI Accelerator pack, including the camera and camera mount, the AI Accelerator and the Polyscope X controller.

Table 1 shows the measured inference times for the different models used in the project, comparing the performance with and without GPU acceleration. The results clearly demonstrate the critical role of the GPU achieving a real-time performance necessary for the project.

Model	Inf. Time (CPU)	Inf. Time (GPU)	Speed-up
Whisper	~22s	~2s	~11x
Qwen3	~5s	~1.4s	~3.57x
YOLOv8	~0.3s	~0.02s	~15x
SAM	~15s	~0.6s	~25x
Total	~42,3s	~4.02s	~10.52x

Table 1: Inference time comparison of each model with and without using the GPU.

5. DEVELOPMENT

The objective is to create an intelligent robotic system in a unified pipeline capable of understanding and acting upon voice commands from a user. The system follows a modular design, where each component performs a specific task including speech-to-text transcription, object detection, object segmentation and controlling the robot based on the previous.

5.1. SPEECH RECOGNITION

The system uses Whisper [4], an advanced multilingual speech recognition model developed by OpenAI, to transcribe the user's spoken commands into text. Its robustness to noise and accents, its ability of transcribing 99 different languages and the option of translating the transcription to English, makes it ideal for this application.

Since the robot lacks a microphone, a dedicated app, shown in Figure 8, was developed for audio recording.

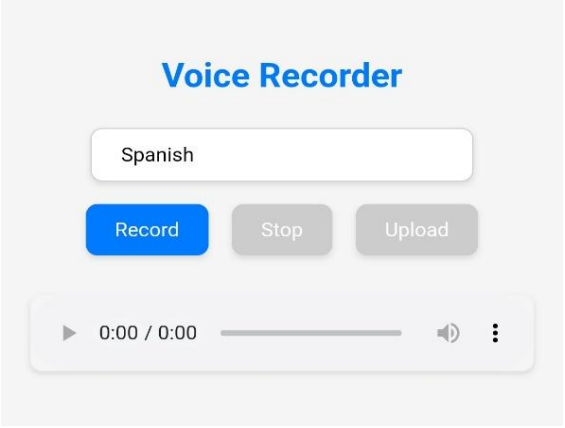

Figure 8: Voice recorder app developed to record audio and send to the pipeline.

The app allows users to record voice commands, such as an audio in Spanish saying "Coge el destornillador" and sends the resulting file to a monitored folder. Since Whisper does not have automatic language recognition, the interface provides a dropdown menu for the user to select the language saving the file using its acronym. For Spanish the audio would be: "es.mp3".

Once the audio is uploaded, the country code is extracted and input to the model, transcribing and translates to English. Returning for the previous example: "Take the screwdriver".

5.2. COMMAND INTERPRETATION

Once the textual command is received, it requires a translation into executable robot commands, which consists of a dictionary with the key "pick" having inside a list with the object.

To achieve this extraction, two different approaches were explored:

- **Sentence transformers:** This approach receives some examples of sentences that it may receive and manages to extract all required pieces of information by comparing it with the provided example sentences. It is very fast but lacks generalization being limited to predefined commands and objects.

- **Small Language Model (SLM):** to generalize the approach and be able to understand a wider variety of sentences, Qwen3 [21] was selected for optimal balance of inference time and accuracy. It is a small language model with 0.6B parameters to which a refined prompt is sent asking to return a dictionary with "pick" key containing the list with the object. Despite being slower, it provides the generalization that was previously lacking, achieving the desired result for a wide variety of sentences.

To compare the SLM's, a test was performed with a set of 50 custom sentences (25 instructing to pick an object, and 25 with other meanings). The results are shown in Table 2 where Qwen3 provided the highest accuracy while having less parameters and maintaining a low inference time.

Model	Nº parameters	Accuracy	Inf. Time
SmolLM2	1.7B	50%	~0.9s
Phi-2	2.7B	10%	~5s
Qwen3	**0.6B**	**75%**	**~0.9s**

Table 2: Comparison between tried SLM models.

So, for the previous example, where the transcription received was "Take the screwdriver" the command that is extracted is:

```
Extracted Actions: {'pick': ['screwdriver']}
```

5.3. OBJECT DETECTION

YOLOv8 [5] is a state-of-the-art real-time object detection model which balances speed and accuracy providing precise and reliable detections while ensuring a high processing speed.

The model detects all relevant objects in the camera's field of view returning their class and bounding box coordinates. This data is parsed to find the object to be picked and return the location of its most precise detection within the image. The detection is performed using the image from the RGB camera, as it is the most similar to the images that were used in the initial training.

5.3.1 DATASET CREATION

Despite the high variety of classes that YOLO offers, it is not aimed for a specific task, for that reason, a small finetuning of the model was performed with different objects to make a demonstration of the capabilities of the proposed project. The objects chosen (see Figure 9) consist of different shapes to show the robustness of the pipeline having a cube, a prism and different cylindrical shaped objects.

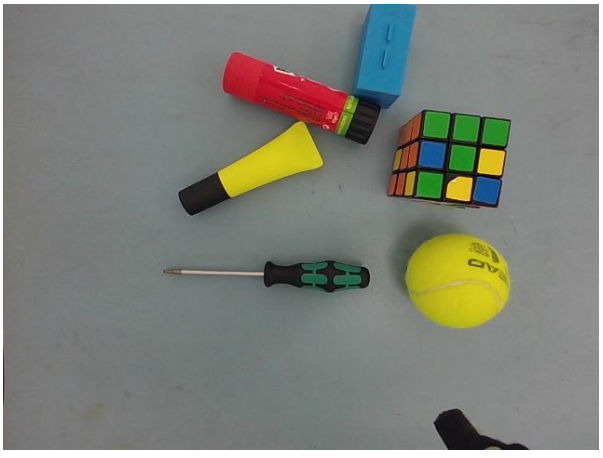

Figure 9: Example of dataset image.

To create a dataset, several images of the selected objects (screwdriver, highlighter, glue, box, ball and Rubik's cube) were captured both individually, only one object in each image, and with scenes with multiple objects present at the same time. This strategy was implemented to address the initial weakness in detection that appeared when training only with datasets from the internet, where the model was only able to detect the object if only one was present in the image, as all those datasets had this common factor. In this way, the model significantly improved its ability to distinguish and identify objects in crowded environments.

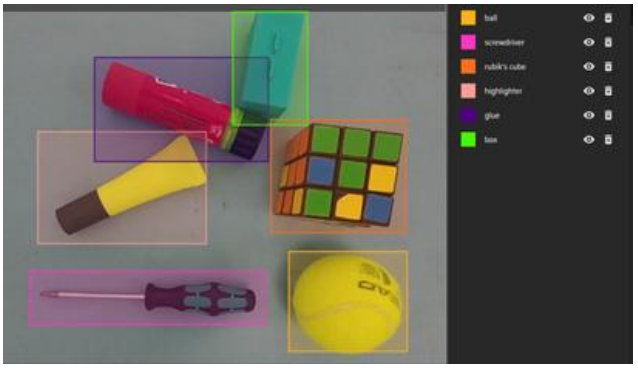

Figure 10: Example of dataset annotation.

The annotation process of the images was performed using the online free tool makesense.ai [16] which facilitated the process of drawing bounding boxes around each object and assign the corresponding class label. The dataset ended up having around 15 pictures of each object individually and the rest

consisted of different combinations of the objects as well as all of them together having at the end a total of 172 image-annotation pairs.

5.3.2 FINETUNING

With the annotated dataset created, YOLOv8 was finetuned to extend its detection capabilities, which already consisted of 80 classes from the COCO [14] dataset, adding the new 6 classes to the already existing ones remapping their ID's.

The model was finetuned for 50 epochs, which provided good results for the new classes without overfitting on them, as tests with several objects of that class were performed and it correctly detected them. To maintain the previous knowledge of the YOLO model the first 10 layers of the network were frozen. This was done as initial layers of a deep learning model detect general features like edges and colours, which are already trained with expertise on the vast COCO dataset. By freezing them, the already learned weights were preserved and the training focused on the deeper layers which are responsible for learning specific features of new objects.

Additionally, a small learning rate (0.003) was set to prevent the previous weights being drastically altered, preserving high performance on original classes and incorporating the new ones. After the training was performed, the model learned to precisely detect and classify all proposed objects even in crowded environments with high precision as shown in Figure 12.

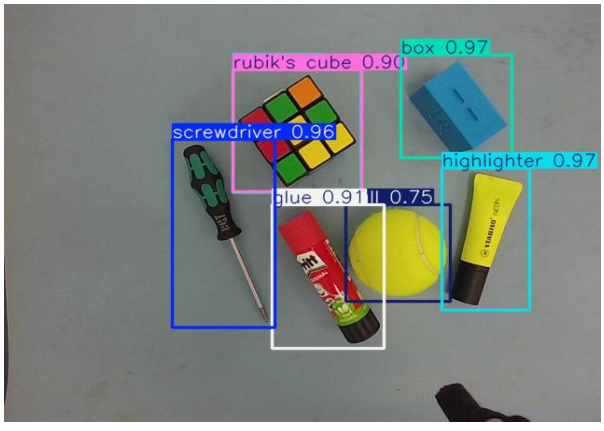

Figure 12: YOLO model detecting all example objects despite being in a crowded environment.

5.4. OBJECT SEGMENTATION

While YOLOv8 provides the bounding box coordinates for the identified object, it is necessary to determine the exact shape of the object within that bounding box to achieve a more precise interaction with the robot. If we used the boxes, the object would have a pick point far from what it could ideally be. As shown in Figure 11, despite still being withing the object shape, it returns an unstable point of the object leading to a missed grasp of the object.

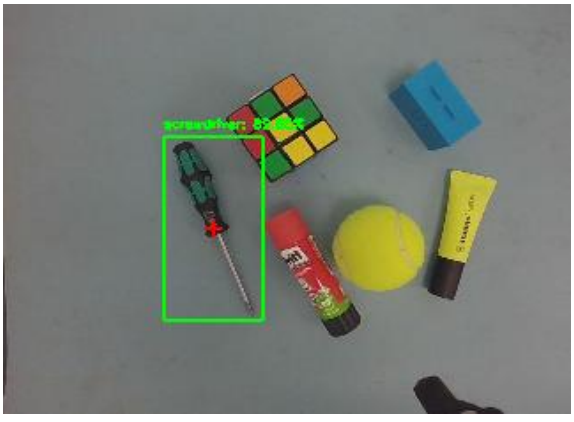

Figure 11: Bounding box center point.

To accomplish this, Segment Anything Model (SAM) [13] has been used to obtain a pixel-wise mask of the object, which precisely delimits the boundaries of the target to pick, removing all the background area that a bounding box provides that could lead to wrong grasping attempts. SAM is a state-of-the-art foundational model for image segmentation whose key advantage lies in its very strong zero-shot generalization, meaning it can segment objects that it has never seen before, under new conditions and domains, without the need for training or fine-tuning on a specific dataset. For all these benefits, SAM has been selected to achieve a highly accurate performance while simplifying the development process.

After detecting the object and extracting the box with YOLOv8, the content inside the box is fed to the SAM predictor, effectively outputting a detailed mask of the object that clearly delimits the contours of it, ensuring the posterior computation of the pick point is always within the object boundaries. As can be seen

in Figure 13, after the box of the object is found, the object is segmented and delimited (shown in red).

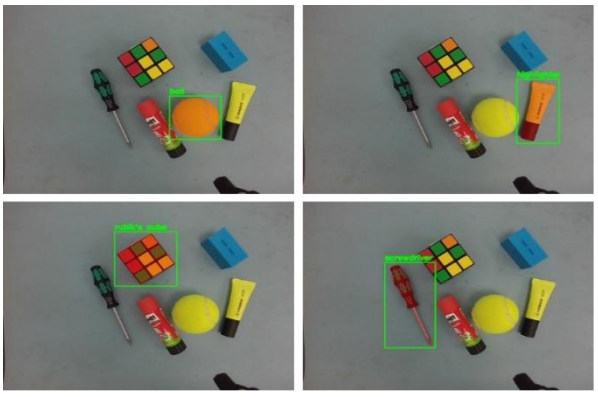

Figure 13: SAM segmentation for the YOLO detections.

Once the object is segmented, the ideal pick point must be computed, as not always picking from the centre is the ideal.

5.4.1 PICK POINT COMPUTATION

Once the object's precise shape is extracted with the SAM segmentation mask, the optimal point and angle within that mask is determined for the robot to grasp the object effectively.

5.4.1.1 CENTROID CALCULATION

The initial candidate for the grasp point is determined using image moments, a feature analysis technique from OpenCV. Image moments are used to calculate the geometric properties of a shape by treating the points of the binary segmentation mask as a single shape, calculating its moments to find its centroid. The centroid is the geometric centre of the mask's pixels, approximating the real "centre of mass" of the object, being an excellent initial candidate for pick point based on the object's overall shape.

5.4.1.2 STABILITY OPTIMIZATION

While the geometric centroid is a valid starting point, it may not always be ideal for physical grasping. For certain objects, as shown in Figure 14, for the highlighter, the centroid is located too close to an edge, potentially leading to an unstable grip. To improve robustness, the system searches for the most optimal pick point along the object's main axis. The purpose of it is to identify a position that is not only stable but also as close to the object's centre of mass as possible. It validates multiple more stable points and selects the closest to the original geometric centroid. This adjustment helps ensure that the gripper targets a more stable point in all cases, shifting it for complicated objects (highlighter), while maintaining (glue) or minimally shifting (screwdriver) improving the overall success rate of the picking task. See Appendix [C] for a more detailed explanation.

5.4.1.3 ORIENTATION CALCULATION

Finally, the grasp orientation is obtained from the geometry of the segmentation mask using Principal Component Analysis (PCA) which identifies the principal axis of variance of the object's shape, being the major axis (shown in yellow) the object's longest dimension, and the minor axis (shown in orange) perpendicular to it (see Figure 14).

The ideal grasp angle is aligned with the minor axis, allowing the gripper to grasp the object across a narrow and more stable width effectively calculating a 2D pixel-space angle.

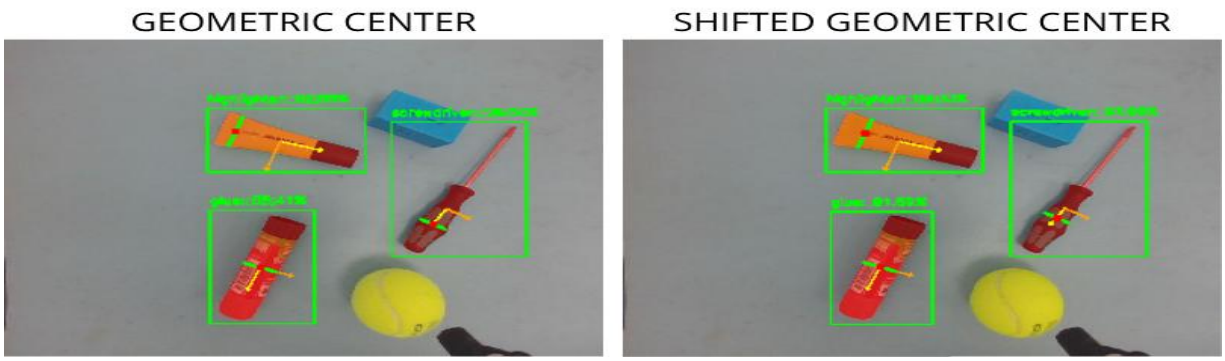

Figure 14: Difference between geometric center and shifted geometric center.

5.5. COORDINATE TRANSFORMATION

To allow the robotic arm to physically interact with an object, the detected 2D pixel coordinates of the pick point must be converted into the robot's 3D coordinate system. This crucial transformation is achieved reprojecting the pixel into the 3D world using the previously obtained calibration parameters. A 3D ray is cast from the camera's origin through the undistorted pixel point, and its intersection with a predefined plane representing the table surface (Z=0 in robot's base frame) determines the final X and Y coordinates for robot's target pose.

Crucially, the Z coordinate is not determined by the depth camera but instead set to a fixed height relative to the known table plane, with a fixed safety offset to define the robot's approach height. This decision was made after initial tests that revealed significant drawbacks when using the depth camera stream which included high latency (approximately a 5-second refresh rate) and insufficient precision, as the sensor struggled to distinguish the object's height from the table surface providing the almost the same distance as if the objects were flat. Therefore, using a fixed plane for the Z-coordinate proved to be a more robust and efficient solution for this application.

5.6. ROBOT COMMUNICATION

Once the final 3D coordinates for the robotic arm are computed, the information must be sent from the external computer that runs the AI models to the UR controller. This connection is done using a socket-based communication protocol over a Transmission Control Protocol/Internet Protocol (TCP/IP) network which creates a robust and direct channel between both parts enabling an almost real-time control.

While the AI Accelerator natively support the Robot Operating System 2 (ROS 2) [22], a more direct communication using TCO/IP was used. Given the time constraints, the primary focus was to develop the core AI pipeline, and adopting ROS 2 would have introduced a significant learning curve and development overhead. A direct TCP/IP socket offered a simpler, yet robust and highly efficient solution for the initial implementation. However, the project's modular architecture has been designed to be well-suited for a future migration to a ROS 2 framework, allowing for more advanced integrations and scalability.

The communication is built on a client-server architecture following this setup:

- TCP/IR Server: The UR robot controller runs its own program that acts as a socket server which opens a specific network port and continuously listens for incoming data being ready to receive and execute commands.

- Python client: The computer where the main script is executed, the AI Accelerator, acts as the client, and once the pick pose is determined it connects to the robot server's IP address and port and sends it.

TCP is a connection-oriented protocol that guarantees reliable and ordered delivery of data which is a critical aspect for robotics. When sending commands with precise coordinates, it is essential that the data arrives intact without any corruption, as a wrong pose could mean colliding with an unwanted object, or even worse, with the person it is meant to be helping. Despite that, one of the reasons UR is the robotics' company contributing with the project, is that they make one of the best collaborative robots in the market, and when collision with a person or object it simply stops any motion being performed to ensure the safety of both the robot and its surroundings.

This socket connection provides a persistent communication link with minimal latency. Once established, the control system can send the poses directly and almost instantly to the controller effectively moving the robot to the expected position to pick the object effectively combining the digital perception with the physical action.

5.7. ROBOT PROGRAM

The robot program starts in a default pose, optimized for camera visibility to detect objects and waits for a message from the socket. When it receives a pose, it moves on top of it, at around 10cm above, to ensure not hitting any object. After that, it goes down for 5cm, this distance is set to match the range that the gripper goes downward when being closed.

After performing the move to the pose that was computed in the computer and going down, the robot closes the gripper, picks the object and goes back up. Then, it rotates into a fixed position to ensure the safety of the camera, as the next move, depending on the pick orientation, could cause a collision with the robot arm.

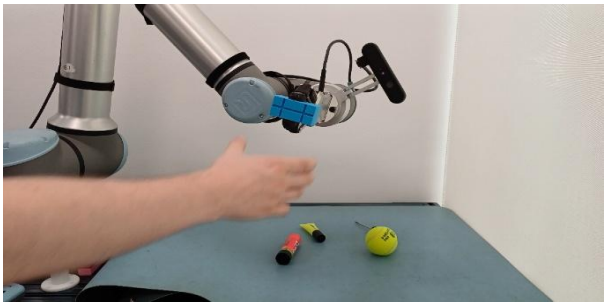

Figure 15: Robot giving the object to the user.

Finally, the object is given to the user by moving to a defined give position and opening the gripper when accepting the "Leave object" popup on the robot controller and goes back to the default position waiting for another object to pick.

6. PERFORMANCE

The performance of the system is evaluated based on speed and task effectiveness. For a human-robot interaction, both aspects are critical to ensure a usable, reliable and real-time experience.

6.1. SPEED

A key consideration is the initial loading time of the AI models. At the start of the pipeline, the system performs a one-time initialization that loads the four models (Whisper, Qwen3, YOLOv8 and SAM) into the GPU's memory. This process is resource-intensive and can take up to 25 seconds, with Whisper and Qwen3 being the models with the highest number of parameters and therefore taking the longest to load. Despite this "long" wait, it is only performed once when the program is launched.

Once the models are loaded, the system operates with high efficiency and near real-time processing. As detailed in table Table 1 in AI ACCELERATOR section, the total inference time for the entire pipeline is approximately 4.02 seconds. This means that after the user uploads a voice command, the system is capable of transcribe the audio, interpret the command, detect the object, segment its shape, calculate the optimal pick point, and send the final pose to the robot in about four seconds. This fast response is fundamental for the system's viability.

6.2. TASK EFFECTIVENESS

Beyond speed, the performance of the project can be measured by its ability to successfully complete the task with a variety of objects. The pipeline was tested with all six custom objects selected to have diverse shapes, sizes and textures.

The effectiveness of the system relies on each module working correctly:

- **YOLOv8 model**: after finetuning, demonstrated high accuracy in detecting and classifying all objects even in cluttered scenes with multiple objects present as shown in the detection images.

- **SAM model**: consistently provided a precise segmentation mask for the detected object, being crucial for the subsequent grasp calculation.

- **Grasp Computation**: computing the geometric centre of the object and using PCA on the object's mask to determine the grip orientation proved to be robust for the tested objects which already correspond to a wide sample of objects that are pickable with the gripper used.

The qualitative testing confirmed a high success rate for picking the objects while maintaining low computational time.

Despite mostly picking objects without flaws, the computed pick point and the actual pick point are not exactly the same due to the unperfect calibration of the camera. This exact example can be seen in the videos where there is a slight mismatch, despite that, this project does not require a millimetric precision which makes this small error admissible for the task.

For a demonstration of the task see videos attached in: https://drive.google.com/drive/folders/19ORpBVpE9-OoYJXkjERhtCLZQgWj0_K2?usp=sharing.

7. FUTURE WORK

The project has established a robust foundation for a voice-controlled robotic assistant, however, there are numerous aspects that could be improved further

enhancing its capabilities providing further aid to users.

7.1. AI ACCELERATOR

The current system uses a socket-based protocol for communication where the robot and the computer work as separate entities. A significant future step would be to use the full potential of the AI Accelerator and transition to a more integrated architecture.

- **ROS 2**: By rebuilding the pipeline using Robot Operating System 2 (ROS 2) which is natively supported by PolyScope X and the AI Accelerator would build a more standardized and scalable framework for managing the data exchange.

- **UR Cap**: Additionally, it could be created an UR Cap (Universal Robots plugin format) to provide a user interface directly on the robot's teach pendant, allowing users to manage the system without a separate computer. NVIDIA Isaac Sim: Using Isaac Sim, a NVIDIA's simulation platform, new algorithms and complex behaviours could be tested before having to test on the physical hardware.

7.2. ENHANCED AUDIO AND COMMANDS

Despite the system can identify a high variety of commands, it only works with one object at a time. An extension of the command interpretation model would allow the user to ask for all the objects it requires without the need to sending an audio for each specific one. Additionally, the audio recognition system could be improved to work like a home assistant, in a similar way that Alexa or Google Home do, which are continuously listening, not requiring the user to specifically send an audio to it.

7.3. ADVANCED GRASPING

A more advanced grasp-planning algorithm could be implemented to improve the success with a wider variety of objects. Instead of using an approach that could be considered "rule based", with the mask's momentum and geometric centroids, a specific model to detect the grasping point of objects could be trained, to improve its performance and truly obtain the most ideal pick point based on a true approximation of the centre of mases by training a model specific for this.

7.4. 3D CAMERA AND DEPTH INTEGRATION

The current system relies on a 2D projection onto a fixed plane to determine the object's position, an approach made to ensure robustness and speed. Future work could focus on fully using the camera capabilities or even changing to a more powerful camera to handle objects of varying heights and positions not constrained to a flat surface.

- **Performance Optimization**: the initial implementation had latency issues with the depth camera refreshing at very low rate. A future task would be optimizing the depth stream processing to handle the depth data, obtaining it without causing a bottleneck.
- **3D grasping:** With an optimized depth stream, the system could use the point cloud generated by the depth data to determine the precise Z-coordinate of the object's top surface, enabling the robot to accurately pick objects of different heights, grasping objects stacked on top of each other or even operate on uneven surfaces increasing the system's flexibility and its possible applications.

8. CONCLUSIONS

This project successfully achieved its primary objective of designing and implementing an autonomous robotic system capable of understanding and executing voice commands to perform pick-and-place as an assistive tool for individuals with reduced mobility. The system moves beyond preprogrammed instructions, enabling users to interact with their environment and manipulate objects by speaking.

By implementing a pipeline of state-of-the-art AI models from different fields, from understanding spoken language to visually analysing objects, this work has demonstrated a viable and powerful solution. A key achievement is the precision of this end-to-end manipulation pipeline. By combining a fine-tuned YOLOv8 for accurate object detection and classification with the pixel segmentation masks from SAM, the system identifies a robust grasp point on the object's 2D image. This point is then transformed into a 3D world coordinate with high positional accuracy, demonstrated by a translational standard deviation of 0.89 from the hand-eye calibration. The entire process

is executed in near real-time, completing in approximately 4 seconds. The final robot pose is sent almost instantly to the robot's controller though a TCP/IP socket, a crucial element that bridges the gap between digital perception and physical action, enabling effective interaction with individuals. The projects modular design not only proved robust for the current implementation but also lays the groundwork for future enhancements, such as migrating to a ROS 2 framework.

In conclusion, this work serves as a successful proof of the potential of combining AI with collaborative robotics. While the system successfully meets all its objectives, it also lays a solid groundwork for future enhancements demonstrating a viable path towards a future of machines that can truly understand and help humans.

REFERENCES

[1] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 4th ed. Pearson, 2020. [Online]. Available: https://aima.cs.berkeley.edu/

[2] R. Murphy, *Introduction to AI Robotics*, 2nd ed. MIT Press, 2019. [Online]. Available: https://books.google.es/books/about/Introduction_to_AI_Robotics.html?id=RVlnL_X6FrwC&redir_esc=y

[3] RecFaces, "What is Voice Recognition? Voice & Speech Recognition Overview." [Online]. Available: https://recfaces.com/articles/what-is-voice-recognition

[4] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, and I. Sutskever, "Robust Speech Recognition via Large-Scale Weak Supervision," *OpenAI*, 2022. [Online]. Available: https://cdn.openai.com/papers/whisper.pdf

[5] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection" *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. [Online]. Available: https://arxiv.org/abs/1506.02640

[6] J. Terven and D. Cordova-Esparza, "A Comprehensive Review of YOLO Architectures in Computer Vision: From YOLOv1 to YOLOv8 and YOLO-NAS,", 2023. [Online]. Available: https://arxiv.org/abs/2304.00501

[7] HuggingFaceTB, *SmolLM2-1.7B-Instruct*. Hugging Face, 2024. [Online]. Available: https://huggingface.co/HuggingFaceTB/SmolLM2-1.7B-Instruct

[8] Universal Robots, "AI Accelerator," *Universal Robots Developer Portal*. [Online]. Available: https://www.universal-robots.com/developer/ai-accelerator/

[9] Wikipedia contributors, "Kanban," *Wikipedia*. [Online]. Available: https://es.wikipedia.org/wiki/Kanban

[10] Orbbec, "Gemini 335 Series – Stereo Vision 3D Cameras." [Online]. Available: https://www.orbbec.com/gemini-335lg/

[11] Universal Robots, *UR10e Robot Datasheet – E-Series*. Universal Robots, 2023. [Online]. Available: https://www.universal-robots.com/media/1807466/ur10e_e-series_datasheets_web.pdf

[12] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, G. Gkioxari, P. Dollár, and R. Girshick, "Segment Anything,", 2023. [Online]. Available: https://arxiv.org/pdf/2304.02643

[13] Meta AI, "Segment Anything," *Segment Anything Project*, 2023. [Online]. Available: https://segment-anything.com/

[14] T-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common Objects in Context," in *European Conference on Computer Vision*, 2014. [Online]. Available: https://arxiv.org/pdf/1405.0312

[15] OnRobot, "RG2 – Flexible 2 Finger Robot Gripper," *OnRobot ApS*. [Online]. Available: https://onrobot.com/en/products/rg2-gripper

[16] P. Skalski, "Make Sense," *makesense.ai*. [Online]. Available: https://www.makesense.ai/

[17] OpenCV Team, *OpenCV*. 2025. [Online]. Available: https://opencv.org/

[18] Z. Jaadi, "A Step-by-Step Explanation of Principal Component Analysis (PCA)," *Built In*, 2021. [Online]. Available: https://builtin.com/data-science/step-step-explanation-principal-component-analysis

[19] IBM, "Sockets," *IBM AIX 7.2 Documentation*, 2024. [Online]. Available: https://www.ibm.com/docs/es/aix/7.2.0?topic=concepts-sockets

[20] Universal Robots A/S, "PolyScope X Software Handbook," *Universal Robots*, 2024. [Online]. Available: https://www.universal-robots.com/manuals/

[21] HuggingFaceTB, *Qwen3-0.6B*. Hugging Face, 2025. [Online]. Available: https://huggingface.co/Qwen/Qwen3-0.6B

[22] Open Robotics, *ROS 2 Documentation*. Open Robotics, 2025. [Online]. Available: https://docs.ros.org/en/jazzy/index.html

APPENDIX

A. PROJECT MANAGEMENT

A1. Table of Objectives

1. Learn to program UR robots
   - 1.1. Complete UR online training modules
     - 1.1.1. Basic e-Serire training
     - 1.1.2. Professional e-Series training
     - 1.1.3. e-Series applications
   - 1.2. Complete the basic and advanced training courses at the UR Academy
2. Speech Recognition model
   - 2.1. Research existing models
   - 2.2. Perform inference
   - 2.3. Connect to app
3. Command Interpretation
   - 3.1. Research existing models
   - 3.2. Perform inference
   - 3.3. Extract command
4. Detection + Classification model
   - 4.1. Research existing models
   - 4.2. Finetune/train model
   - 4.3. Perform inference
   - 4.4. Connect to camera
5. Segmentation
   - 5.1. Research existing models
   - 5.2. Perform inference
   - 5.3. Connect to camera
6. Pick Point Selection
   - 6.1. Explore different methods
   - 6.2. Approximate centre of masses
   - 6.3. Compute robot pose based on pick point
7. Robot
   - 7.1. Define end effector/s (gripper, vacuum…) to be used.
   - 7.2. Install necessary UR Caps
   - 7.3. Calibrate camera
   - 7.4. Convert detected coordinates to robot coordinates
   - 7.5. Join to single program
   - 7.6. Make final demo/app

Table 3: Objectives to be accomplished

A2. Table of Tasks

Planning			
Task	**Description**	**Start**	**End**
Initial Meeting (20/02/2025)			
Learning		6/02/2025	15/03/2025
1	UR Online Training	6/02/2025	7/02/2025
2	UR Basic/Advanced Training	10/02/2025	13/02/2025
3	Get familiar with ROS2	17/02/2025	26/02/2025
4	Get familiar with and complete AI Accelerator Demos	17/02/2025	14/03/2025
Initial Report Delivery + 1st Mandatory Follow-up (17/03/2025 – 21/03/2025)			
Development		16/03/2025	2/06/2025
5	Research Speech Recognition (SR) models	16/03/2025	16/03/2025

6	Research Detection/Classification (Det/Class) models	17/03/2025	17/03/2025
7	Test SR and perform inference	18/03/2025	21/03/2025
8	Test Det/Class and perform inference (including annotation and finetune if needed)	22/03/2025	4/04/2025
9	Define end effectors, cameras for the robot and objects	7/04/2025	11/04/2025
10	Install URCaps and calibrate camera/s	14/04/2025	18/04/2025
11	Connect models to robot	21/04/2025	25/04/2025
12	Convert detection coordinates to robot coordinates	28/04/2025	2/05/2025
13	Report the progress for the final report	21/03/2025	4/05/2025
Progress Report Delivery + 2nd Mandatory Follow-up (5/05/2025 – 9/05/2025)			
14	Perform previous steps within the AI Accelerator	2/05/2025	2/06/2025
Deployment		3/06/2025	9/07/2025
15	Join To single Program	3/06/2025	9/06/2025
16	Prepare final demo/app	10/06/2025	16/06/2025
17	Final report	10/05/2025	23/06/2025
Final Report Delivery + 3rd Mandatory Follow-up (23/06/2025 – 27/06/2025)			
Final Report Submission (29/06/2025)			
18	Prepare final presentation	28/06/2025	2/07/2025
Final Presentation (3/07/2025 – 9/07/2025)			

Table 4: Table of tasks

A3. GANTT

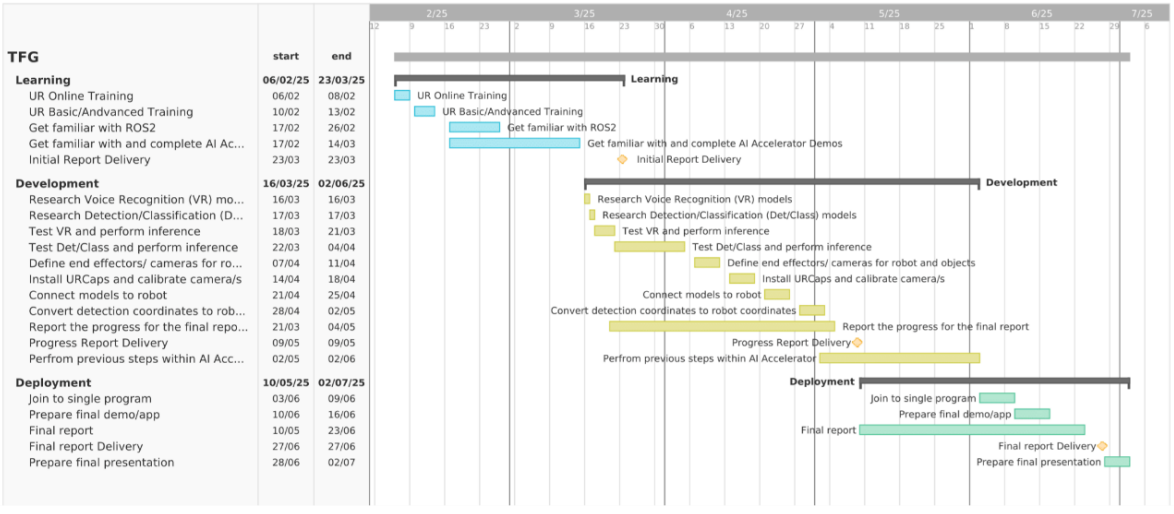

Figure 16: Gantt chart of the project

B. CAMERA CALIBRATION

B1. Intrinsics and hand-eye calibration

A calibration process is usually divided into two parts: obtaining the camera's intrinsics and performing the hand-eye calibration.

- **Intrinsics Calibration:** Defines the camera's internal optical characteristics. It analyses the chessboard images to compute the intrinsic matrix (K) and the distortion coefficients.
    - **K**: Is a 3x3 matrix containing the camera's internal optical properties, such as the focal length (fx, fy) and the optical center (cx, cy). This matrix is essential for mapping 3D points from the camera's coordinate system onto the 2D image place.
    - **Distortion Coefficients**: Are set of values that model the distortion introduced by the camera lens. These values are used to correct image points, ensuring that straight lines in the real world appear as straight lines in the image, a crucial aspect to obtain accurate measurements.
- **Hand-eye calibration:** Is the core of the process, which obtains a fixed transformation between the robot flange's frame and the camera's coordinate frame. This transformation is obtained by matching the series of robot flange poses and the calibration board relative to the camera solving the AX = XB equation to find the constant transformation X that goes from the robot's flange (A) to the calibration board (B). X is a 4x4 homogeneous matrix that defines the precise 3D rotation and translation from the camera to the robot flange.

B2. Data collection and filtering

A custom script using the OpenCV library was used to capture the image-pose pairs. However, initial calibration attempts revealed inconsistencies despite using valid image, the resulting transformation was inaccurate causing computed poses to be far from their real-world locations.

To solve this, a larger set of image-pose pairs was taken to ensure grater variety of viewpoints, which allowed for the selection of a high-quality subset of frames. Then a script was developed to perform an initial calibration using all frames and discard the "outliers" based on two metrics before recalibrating.

- **Reprojection Error:** Measures the pixel distance between a detected chessboard corner and the position where the camera optical model predicts it should be. Frames with high reprojection error were discarded as they lead to an inaccurate intrinsic optical model.
- **Hand-Eye Consistency:** Evaluates how well each frame satisfies the AX = BX equation by checking if the robot's movement (A) accurately corresponds to the camera's observation of the board (B) given the current transformation X.

Based on these metrics, any image-pose pair whose error exceeded 0.5 standard deviations from the mean was discarded. Through this process, the initial 81 images were filtered down to a subset of 34 optimal frames. This refinement drastically improved the hand-eye consistency, reducing the mean $||AX - BX||$ error from an initial value of ~13 to a final value of ~1.

B3. Analysis of calibration results

An extra validation was performed to check the stability of the transformation. This consisted of calculating the 3D pose of the chessboard from each of the 34 final frames, ideally achieving identical results with translation and deviation close to 0.

The validation of the hand-eye showed a significant discrepancy between translational and rotational accuracy. The system demonstrated excellent translational stability with a standard deviation of only 0.89mm, but very poor rotational stability with a standard deviation of 88º. A detailed analysis of the likely causes is performed in this section.

The primary reason for poor rotational accuracy in "AX = BX" hand-eye calibration is often a poor dataset. To accurately obtain the rotational component of the matrix transformation X, the dataset of robot movements (A matrices) must contain varied rotations of the robot's end-effector around multiple axes. While the 81 poses covered a wide range of positions, they likely did not include enough distinct rotational movements, even when 30 of the 81 images were taken focusing only on rotation of the end-effector. So, the reason could be the dataset contained mainly translational movements (moving in X, Y or Z without changing the orientation) providing a good calibration for translation but not for rotation.

Another possible factor is the camera's hardware characteristics. The Orbecc Gemini 335Lg is an active stereo camera that uses infrared (IR) projectors to cast dot patterns onto a scene to calculate depth with high precision. While the calibration only uses the RGB sensor, a small amount of this intense infrared light can leak into the RGB sensor introducing a systematic noise patter that slightly biases the calibration board find algorithm at sub-pixel level. To prove the hypothesis, the camera lens was covered with a black cloth blocking all external light which clearly revealed the IR leakage. The results are shown in Figure 17, where this leakage is clearly visible both in the raw (black) image, and in the histogram equalized image, confirming the presence of a

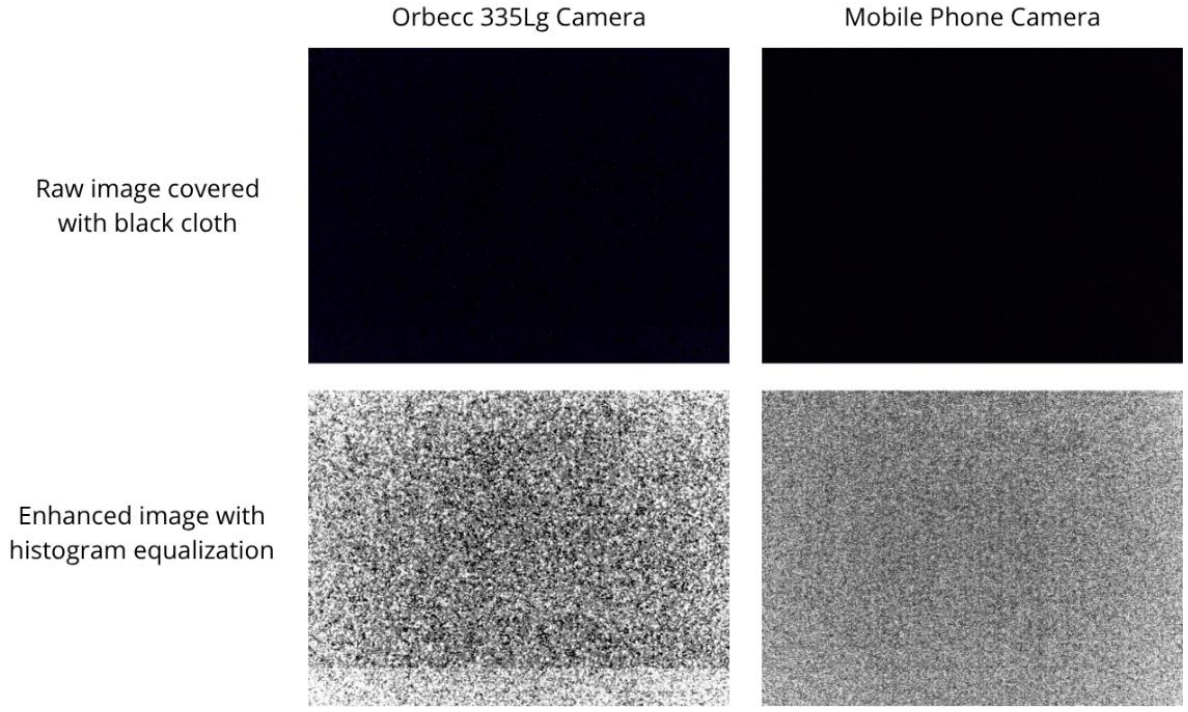

Figure 17: Comparison with noise between the project's camera (left) and a standard mobile phone camera (right) under complete darkness and when equalizing its histograms.

systematic noise pattern unique to the Orbecc camera. When comparing with the mobile phone camera, the raw image appears fully black and when making the histogram equalization it only produced random electronic noise.

This type of measurement noise has a much greater effect on the rotation calculation, as it depends on the precise relative geometry of all detected corners, than on translation, where small errors are averaged out when determining the board's centre. This hardware induced noise combined with a low rotational variety dataset provides a plausible explanation for the observed results.

C. PICK POINT COMPUTATION ALGORITHM

This section provides a detailed explanation of algorithm used to determine the optimal grasp point and angle from the SAM segmentation mask.

C1. Image Moments and Centroid Calculation

Image moments is a computer vision technique for calculating statistical properties of a shape. In physics, "moments" describe how the mass of an object is distributed; in an image, the same concept is applied to the pixels of a shape. For a binary segmentation mask, the objects belonging to the object (white pixels) have a uniform value (or "mass") of 1, while the background pixels (black) have a value of 0.

- **Zeroth moment**: Represents the total area of the object in pixels. It is calculated by summing the value of all pixels in the mask. Since object pixels are 1 and background is 0, the sum is the total number of white pixels. This value is used to normalize the other moments.

- **First order moments**: The moments are used to find the centroid of the object, which is the geometric centre or "centre of mass". They are calculated as the weighted average of pixel coordinates.

    - **X-Moment**: Is the sum of all x-coordinates of the object's pixels, measuring the distribution of the object's "mass" along the horizontal axis.

    - **Y-Moment**: Is the sum of all y-coordinates of the object's pixels, measuring the distribution along the vertical axis.

The centroid coordinates are then calculated by dividing the first-order moments by the total area (zeroth moment) resulting in the geometric centre of the object's 2D silhouette.

C2. Pick point shift

While the geometric centroid is a good starting point, it is not always the most stable, especially for objects with irregular or non-convex shapes. To find a more robust grasp point, the system modifies the centroid searching a more stable position along the object's principal axis. The algorithm that makes this shift does the following:

1. **Principal Axis**: first, the algorithm analyses the object's shape using the PCA to identify the object's major axis, being the line that represents the longest dimension of the object.
2. **Candidates point search**: then a better pick point is search along this axis iterating through multiple candidate points moving away from the centroid.
3. **Grasp validation:** for each candidate, a grasp line perpendicular to the axis is simulated, representing the line between the gripper's fingers and calculate the percentage of this line that falls within the object's mask.
4. **Solidity check:** a grasp is considered valid if the overlap exceeds a certain threshold, in this case 90%. This ensures that the gripper will have a firm hold across the object's body rather than attempting an unstable grip on a narrow edge or corner.
5. **Optimal point selection:** from all valid grasp points, the closest to the original geometric centroid is selected ensuring the grasp point is as stable as possible while being near the geometric centre effectively shifting the pick point from a potentially unstable geometric centre to a more practical and robust position.