ECE 445

SENIOR DESIGN LABORATORY

DESIGN DOCUMENT

Grasping Any Object with Robotic Arms with Language Instructions

---

Team #10

JUNZHOU FANG
(junzhou5@illinois.edu)
JUNSHENG HUANG
(jh103@illinois.edu)
ZIXIN ZHU
(zixinz6@illinois.edu)
ZIXUAN ZHANG
(zixuan21@illinois.edu)

TA: Tianci Cai, Tielong Tang

April 7, 2025

Contents

1 Introduction

1.1 Problem

Individuals who are bedridden or have significantly limited mobility due to conditions such as stroke, spinal cord injuries, or neuro degenerative diseases often face substantial challenges in performing routine daily activities. Tasks as simple as reaching for a water bottle, retrieving medication, or accessing personal devices become daunting and frequently necessitate assistance from caregivers. This dependency can lead to a diminished sense of autonomy and increased strain on both patients and healthcare providers.

The integration of assistive robotic technologies has shown promise in enhancing the quality of life for individuals with mobility impairments. Studies have demonstrated that such technologies can effectively reduce the physical burden on caregivers and improve patient outcomes. [1], [2] Furthermore, advancements in artificial intelligence have facilitated the development of robots capable of assisting the elderly in daily activities, thereby promoting independence and safety.

Despite these advancements, existing assistive devices often lack the capability to interpret natural language commands and adapt to the dynamic environments typically found in bedside settings. This limitation underscores the need for an intelligent robotic assistant that can comprehend voice instructions, accurately identify and locate a variety of objects in real-time, and safely interact with them within the confined space of a patient's bedside. It requires the seamless integration of advanced computer vision, natural language processing, and precise robotic control to perform accurate grasping actions according to oral instruction. Many existing robotic systems are limited to specific contexts or require extensive retraining for new objects, lacking the generalization needed for a broad object vocabulary. Overcoming these hurdles is essential to creating adaptable robotic systems that facilitate patience's independence in dynamic, human-centric settings.

1.2 Solution

Our expected solution is a smart robotic arm equipped with a well-designed recognition system based on computer vision and natural language processing. The image analysis module of the robotic arm will be trained using RGB images and corresponding captions, allowing it to categorize objects in the robotic arm camera. We will also use a language processing module to extract the name of the intended objects from the input text or voice command. Then, the robotic arm will move along the path to grab the object to the designated position. The visual illustration of our robotic arm is shown below.

1.3 High-level Requirements List

- Reliability: The system should maintain a high level of reliability, such as the accuracy of 80% recognition when matching input instructions and figures.

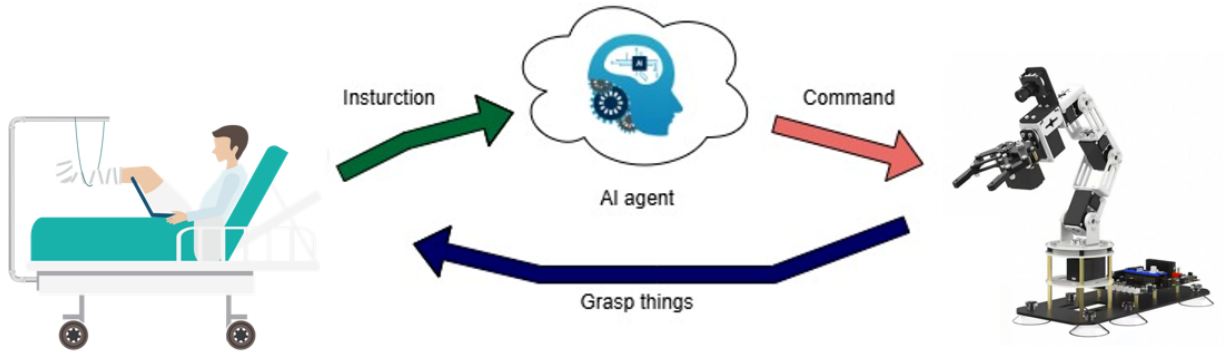

Figure 1: The Visual Illustration

- Generalization: The system should support grasping for at least 10 distinct objects and be able to deal with out-of-vocabulary phenomena.
- Efficiency: The system should avoid collisions during the path execution and complete each grasping task in 2 minutes.

2 Design

2.1 High-level Block Diagram

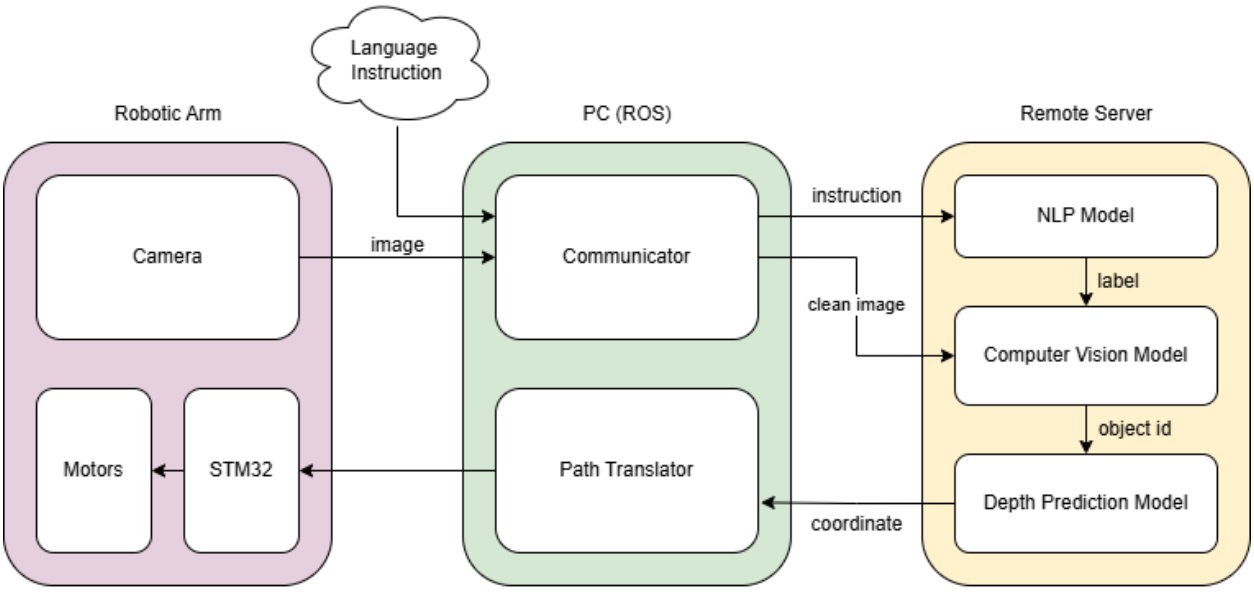

Figure 2: Model Layout

2.2 Robotic Arm

We use the YahBoom DOFBOT SE Robotic Arm, which is driven by an STM32 controller and uses a virtual machine as the master to generate control decisions. This robotic arm has 6-degree-of-freedom serial bus servo and is controlled by the ROS operating system. By installing a microphone and a USB camera on the outside of the robotic arm, we give the robotic arm visual and auditory perception capabilities. For the end effector, we design a mechanical claw with a maximum opening width of 6 cm and a maximum load of 200 g, so that it can grasp common small objects. Figure 3 shows the specifications of this robotic arm [3].

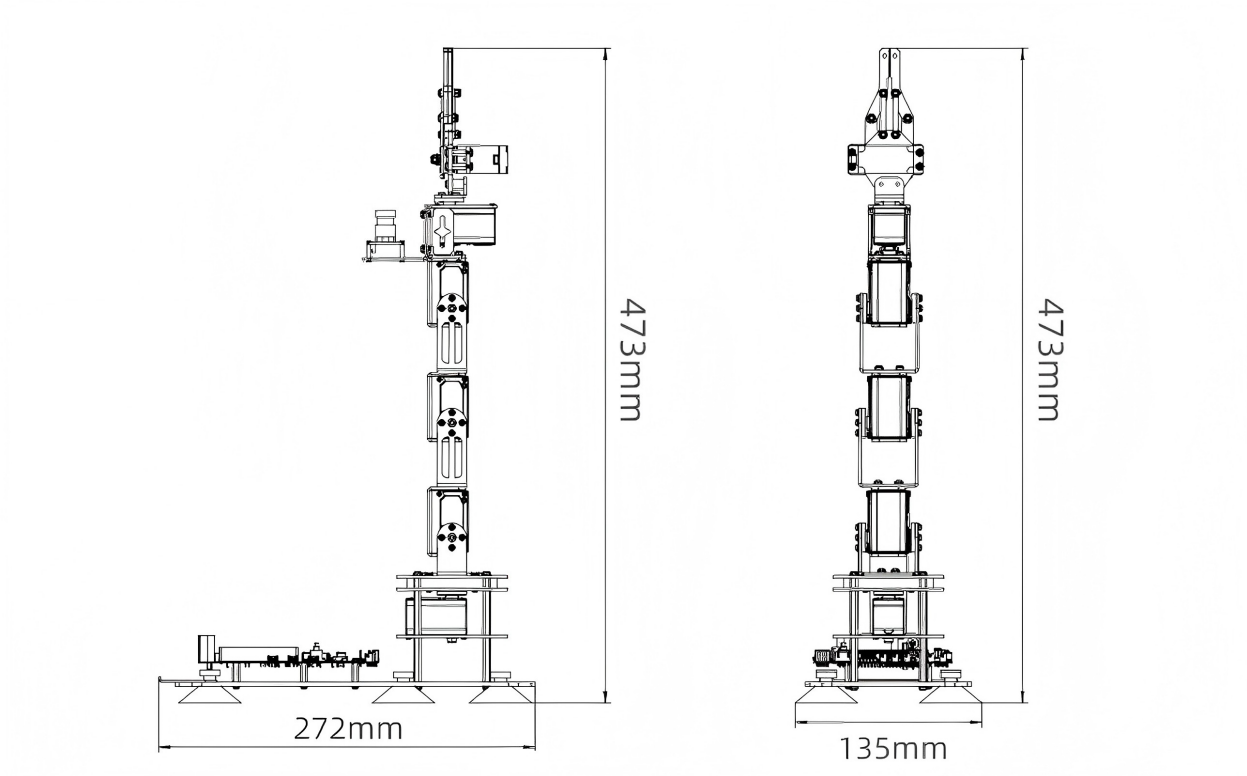

Figure 3: The Specifications Of The Robotic Arm

2.2.1 Camera

The Camera subsystem runs continuously, capturing images once every second, and providing real-time visual data to the Image Encoder subsystem for further processing. By operating in a continuous manner, the system can persistently collect and analyze visual information, facilitating rapid detection of changes or anomalies in the environment. Moreover, the steady one-frame per second capture rate ensures a stable and timely flow of image data, catering to the requirements of subsequent algorithms and monitoring tasks in a broad range of applications.

2.2.2 Microphone

The microphone serves as the input device for capturing voice commands. It enables the robotic arm to receive instructions through natural language. The captured audio is processed by a speech-to-text engine, allowing the system to identify the target object name and intended action. This module enhances the human-robot interaction, making the operation more intuitive and hands-free.

2.2.3 STM32

The STM32F103C8T6 microcontroller serves as the control hub of the robotic arm system. Based on an ARM Cortex-M3 core running at 72 MHz, it integrates 64 kB of Flash memory and 20 kB of SRAM, providing sufficient computational power for real-time control. It features multiple USARTs, PWM-compatible 16-bit timers, and GPIO ports for precise coordination of the six smart servo motors. With built-in USB, I²C, SPI, and ADC interfaces, the STM32 manages sensor inputs, actuator commands, and peripheral communication efficiently, enabling stable and responsive operation of the robotic arm.

2.2.4 Motor

The robotic arm is driven by six DS-SY15A smart servo motors, which are responsible for executing precise and smooth movements based on the planned trajectory computed by the control algorithm. These motors operate at 6.0–7.4V, deliver a rated torque of 15 kgf·cm, and reach a maximum rotation angle of 300°, allowing the arm to execute smooth and flexible movements. With a no-load speed of less than 0.24 sec/60° and metal gear construction, they ensure both responsiveness and durability. Communication is handled via UART serial protocol at 115200 bps, enabling coordinated multi-motor control through the STM32 microcontroller.

2.2.5 End Effector

Since the maximum opening width of the original robot arm end effector was too small, we redesigned the end effector, and its CAD drawing is shown in Figure 4. By laser cutting the acrylic plate, we obtained and assembled the eight parts of the end effector, and the finished product is shown in Figure 5. The optimized end effector has a maximum opening width of 6 cm and a maximum load capacity of 200 grams, which is more suitable for grabbing various daily small objects. It is controlled by six servo motors and works in conjunction with the robot arm's motion planning system.

2.3 PC (ROS)

ROS (Robot Operating System) is an open-source framework that supports the development of robot software. Rather than being a traditional operating system, ROS functions as a middle platform running on top of existing systems like Linux. It provides essential components such as hardware abstraction, low-level device control, inter-process communication, development tools, algorithm libraries, and visualization utilities. One of

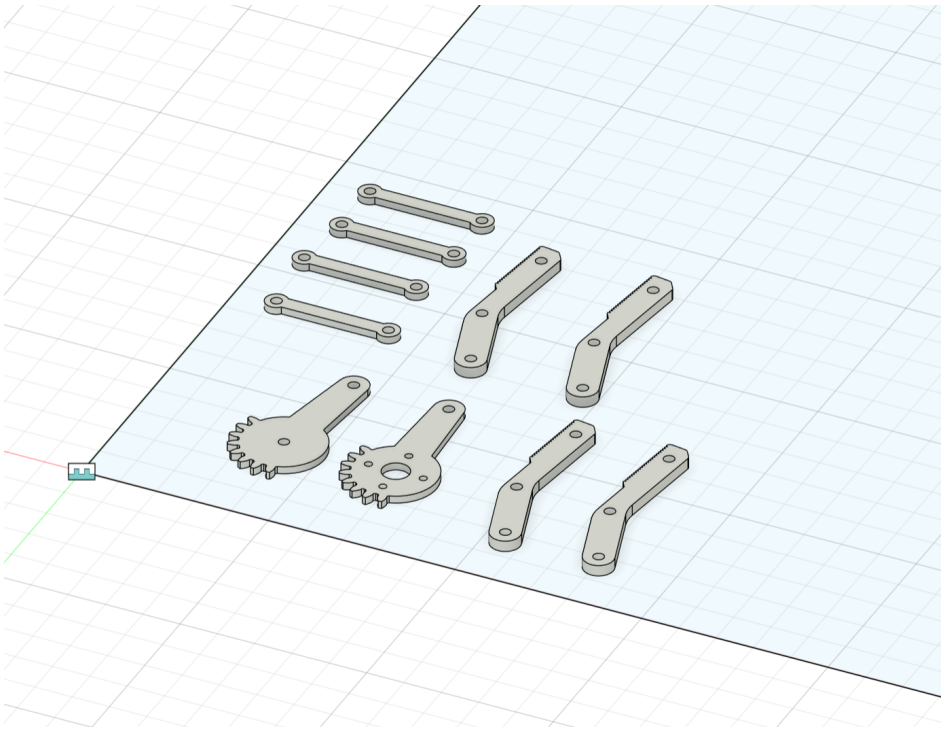

Figure 4: CAD Drawing of Redesigned End Effector

Figure 5: Finished Product of Redesigned End Effector

ROS's key strengths lies in its modular architecture, where different functional components—called "nodes"—communicate through a publish/subscribe messaging mechanism. This enables flexible, distributed development and clean separation of functionalities. Developers can also take advantage of a wide range of open-source packages provided by ROS, including modules for navigation, SLAM, path planning, and image processing, allowing rapid prototyping of complex robotic systems. ROS is widely used in service robots, mobile robotics, industrial automation, UAVs, and many other areas, making it a cornerstone of modern robotics development.

Subsystem Control

For control, we will go to inverse kinematics. After the image processing get the coordinates and orientation of the target, relative to the robotics $(\alpha, \beta, \gamma, x, y, z)$, the transformation will become:

$$
{}^{0}_{E}T = \begin{bmatrix} c_\alpha c_\beta & c_\alpha s_\beta s_\gamma - s_\alpha c_\gamma & c_\alpha s_\beta c_\gamma + s_\alpha s_\gamma & x \\ s_\alpha c_\beta & s_\alpha s_\beta s_\gamma + c_\alpha c_\gamma & s_\alpha s_\beta c_\gamma - c_\alpha s_\gamma & y \\ -s_\beta & c_\beta s_\gamma & c_\beta c_\gamma & z \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

And

$$
{}^{0}_{E}T = {}^{0}_{1}T^{1}_{2}T \cdots
$$

The Denavit–Hartenberg (D-H) parameters provide a standardized method to describe the geometric relationship between adjacent links and joints of a robotic manipulator. This method is widely used to model the forward kinematics of robotic arms.

In the standard D-H convention, the relative transformation between two consecutive coordinate frames is characterized by four parameters:

- θ_i: Joint angle – the rotation about the z_{i-1} axis (variable for revolute joints).
- d_i: Link offset – the translation along the z_{i-1} axis (variable for prismatic joints).
- a_i: Link length – the translation along the x_i axis (a constant).
- α_i: Link twist – the rotation about the x_i axis (a constant).

By using these four parameters, a 4×4 homogeneous transformation matrix can be constructed to describe the pose of each link relative to its predecessor. Chaining these transformations allows for the complete forward kinematic model of the robot to be built.

For transformation ${}^{i}_{i-1}T$, we use D-H parameters to determine which is

$$
{}^{i}_{i-1}T = [Z_{i-1}][X_i]
$$

$$[Z_i] = \begin{bmatrix} \cos\vartheta_i & -\sin\vartheta_i & 0 & 0 \\ \sin\vartheta_i & \cos\vartheta_i & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$[X_i] = \begin{bmatrix} 1 & 0 & 0 & r_{i,i+1} \\ 0 & \cos\alpha_{i,i+1} & -\sin\alpha_{i,i+1} & 0 \\ 0 & \sin\alpha_{i,i+1} & \cos\alpha_{i,i+1} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The transformation matrix T is a 4×4 homogeneous transformation matrix, widely used in robotics and rigid body kinematics to describe the pose (position and orientation) of one coordinate frame relative to another.

It has the general form:

$$T = \begin{bmatrix} \mathbf{R} & \mathbf{p} \\ \mathbf{0}^T & 1 \end{bmatrix}$$

where:

- $\mathbf{R} \in \mathbb{R}^{3\times3}$ is the rotation matrix, representing orientation.
- $\mathbf{p} \in \mathbb{R}^{3\times1}$ is the position (translation) vector.
- The bottom row $[0\ 0\ 0\ 1]$ allows translation and rotation to be combined in a single matrix operation.

Use in Robotics

In robotics, the transformation matrix is used to relate different coordinate frames along the manipulator. For instance, the transformation from frame $i-1$ to frame i is represented as ${}^{i}_{i-1}T$. Then, a point P can be transformed as:

$$P_i = {}^{i}_{i-1}T \cdot P_{i-1}$$

where P_{i-1} and P_i are the homogeneous coordinates of the same point expressed in frames $i-1$ and i, respectively.

Constructing T using D-H Parameters

When using Denavit–Hartenberg (D-H) parameters, the transformation matrix between two successive frames is given by:

$$
{}^{i}_{i-1}T = \begin{bmatrix} \cos\theta_i & -\sin\theta_i\cos\alpha_i & \sin\theta_i\sin\alpha_i & a_i\cos\theta_i \\ \sin\theta_i & \cos\theta_i\cos\alpha_i & -\cos\theta_i\sin\alpha_i & a_i\sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

After writing down the D–H parameter, we can identify each α, ϑ by combining the transformation matrix and comparing them to ${}^{0}_{E}T$. Therefore, those pairs of (α, ϑ) can form a joint space

$$
q = \begin{bmatrix} q_1 \\ q_2 \\ \vdots \end{bmatrix}
$$

Each q_i represents the orientation of each motor. And these will be packed as "Actuator_angle" message and sent to actuator.

Furthermore, to evaluate the velocity and acceleration of bodies. The velocity of body i with respect to body j can be represented in frame k by the matrix

$$
W_{i,j(k)} = \begin{bmatrix} 0 & -\omega_z & \omega_y & v_x \\ \omega_z & 0 & -\omega_x & v_y \\ -\omega_y & \omega_x & 0 & v_z \\ 0 & 0 & 0 & 0 \end{bmatrix}
$$

Where ω is the angular velocity of body j with respect to body i. All the components are expressed in frame k; v is the velocity of one point of body j with respect to body i (the pole). The pole is the point of j passing through the origin of frame i.

The acceleration matrix can be defined as the sum of the time derivative of the velocity plus the velocity squared:

$$
H_{i,j(k)} = \dot{W}_{i,j(k)} + W^2_{i,j(k)}
$$

The velocity and the acceleration in frame i of a point of body j can be evaluated as

$$\dot{P} = W_{i,j}P$$
$$\ddot{P} = H_{i,j}P$$

And

$$\dot{M}_{i,j} = W_{i,j(k)}M_{i,j}$$
$$\ddot{M}_{i,j} = H_{i,j(k)}M_{i,j}$$

Velocity and acceleration matrices add up according to the following rules:

$$W_{i,k} = W_{i,j} + W_{j,k}$$
$$H_{i,k} = H_{i,j} + H_{j,k} + 2W_{i,j}W_{j,k}$$

The absolute velocity is the sum of the parent velocity plus the relative velocity; for the acceleration, the Coriolis' term is also present.

The components of velocity and acceleration matrices are expressed in an arbitrary frame k and transform from one frame to another by the following rule:

$$W_{(h)} = M_{h,k}W_{(k)}M_{k,h}$$
$$H_{(h)} = M_{h,k}H_{(k)}M_{k,h}$$

2.3.1 Communicator

The Communicator module in the ROS-based PC system is responsible for handling data exchange between the ROS environment and external components, such as the STM32 microcontroller and robotic arm subsystems. It acts as a messaging bridge, ensuring that data—whether sensor readings, control commands, or feedback signals—is accurately received, processed, and dispatched across relevant ROS nodes. Using protocols like MQTT or ROS's native messaging interface, the Communicator ensures low-latency, reliable, and real-time communication, which is critical for coordinated robotic actions. It also handles error checking and message formatting, maintaining synchronization between the distributed subsystems. The modularity of ROS allows the Communicator to be reused or extended across different robot configurations without altering the core logic.

2.3.2 Path Translator

The Path Translator module serves as the interface between high-level path planning and low-level actuation commands. It receives planned trajectories, typically expressed in terms of Cartesian or joint-space waypoints, and converts them into executable commands that match the specific control format expected by the robotic arm. This includes

interpolating smooth motion curves, adjusting speeds, and encoding commands into formats supported by the STM32 and motor controllers. The Path Translator ensures that each step of the path is feasible, optimized for efficiency, and respects constraints like joint limits and physical safety. Integrated into the ROS framework, the module works seamlessly with planning nodes and execution nodes, making it a key component in the overall motion control pipeline.

2.4 AI Models

Our AI agent integrates natural language understanding with visual perception to enable precise object manipulation. The agent accepts either text input or speech commands, intelligently identifying objects for robotic arm grasping task.

2.4.1 NLP Model

The primary function of this natural language processing (NLP) model is to transform user-provided instructions (regardless of input text or speech signals). These labels serve as triggers for the YOLOE (CV model) to perform designated tasks. This module acts as a critical bridge between language understanding and visual perception in a multi-modal system, enabling instruction-driven control via natural language. Specifically, this model would be divided into two components.

The first component is the Automatic Speech Recognition (ASR) module, which is responsible for processing incoming audio signals. This module would first improve speech quality through noise reduction and signal enhancement. Then, The refined audio is transcribed into its corresponding textual representation, thereby serving as the input for subsequent natural language processing modules. We choose a lightening framework from Meta AI: Wav2Vec 2.0 [4]. It is a self-supervised speech recognition framework that learns powerful speech representations directly from raw audio using Transformer architectures and contrastive learning. Figure 6 illustrates the workflow of the framework.

The second component is the Semantic Label Extraction module, which processes the textual instructions obtained from the ASR module or directly from the user. It employs predefined regular expression rules to extract key terms that correspond to the target object for grasping, serving as the label for the downstream computer vision module. Recognizing the limitations of rule-based keyword extraction in handling the diversity and ambiguity of natural language, we also consider leveraging semantic similarity models to infer potential target labels. This hybrid approach enhances the robustness and generalization of instruction interpretation.

2.4.2 YOLOE (CV Model)

We primarily select YOLOE as our visual detection model. After ROS reads the image captured by the camera, the image will be inputted into this visual model. The model is responsible for recognizing all objects in the image, by extracting them from raw image

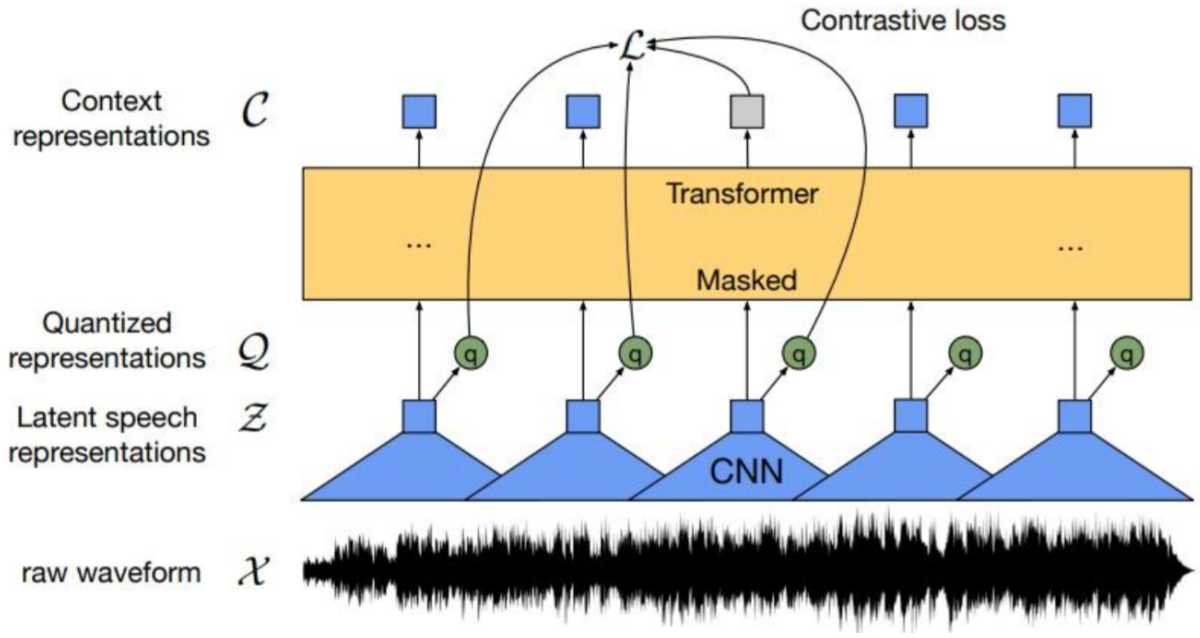

Figure 6: Illustration of Wav2Vec 2.0 framework

and comparing them with pre-defined vocabulary base. Since we already get the instruction from NLP model, we produce a text prompt and just ask the model to select the target object accordingly. The output of the model will be the index of the object. This index along with the raw image will be inputted to the Depth Prediction model to generate the 3D coordinate of the target object.

Model	**Size**	**Params**	**AP**	$\mathbf{AP_r}$	$\mathbf{AP_c}$	$\mathbf{AP_f}$
YOLOE-v8-S	640	13M	27.9 / 26.2	22.3 / 21.3	27.8 / 27.7	29.0 / 25.7
YOLOE-v8-M	640	30M	32.6 / 31.0	26.9 / 27.0	31.9 / 31.7	34.4 / 31.1
YOLOE-v8-L	640	50M	35.9 / 34.2	33.2 / 33.2	34.8 / 34.6	37.3 / 34.1
YOLOE-11-S	640	12M	27.5 / 26.3	21.4 / 22.5	26.8 / 27.1	29.3 / 26.4
YOLOE-11-M	640	27M	33.0 / 31.4	26.9 / 27.1	32.5 / 31.9	34.5 / 31.7
YOLOE-11-L	640	32M	35.2 / 33.7	29.1 / 28.1	35.0 / 34.6	36.5 / 33.8

Table 1: YOLOE version Comparison

We deploy YOLOE-11m to the server with two concerns. The first one is the requirement of our task. Given a near-bed, medical-aid scenario, input images will have a relatively clean background and contain commonly-seen objects. Therefore, we prioritize the model's accuracy while accepting a sub-optimal ability in categorizing messy or unseen objects. The other one is the cost of the model. We ask the model to response in a short period of time after the instruction is inputted. Thus, the complexity of the model has

to be limited. Given these two concerns, YOLOE-11m achieves an optimal trade-off between performance and size, with only 27M parameters for inference. However, since the choice of visual model involves a seamless integration to other modules, we may change the model version in later implementation. A complete versoin table of YOLOE can be referred in Table 1. [5]

2.4.3 Depth Prediction Model

The depth prediction module employs the DepthFM [6] model as its core architecture, aiming to transform monocular camera images into depth-enhanced representations. In the preprocessing stage, the module performs essential image refinement operations, such as pixel completion and frame rate enhancement, to improve spatial continuity and visual quality. The refined images are then fed into DepthFM to produce per-pixel depth maps, providing spatial awareness for downstream visual tasks.

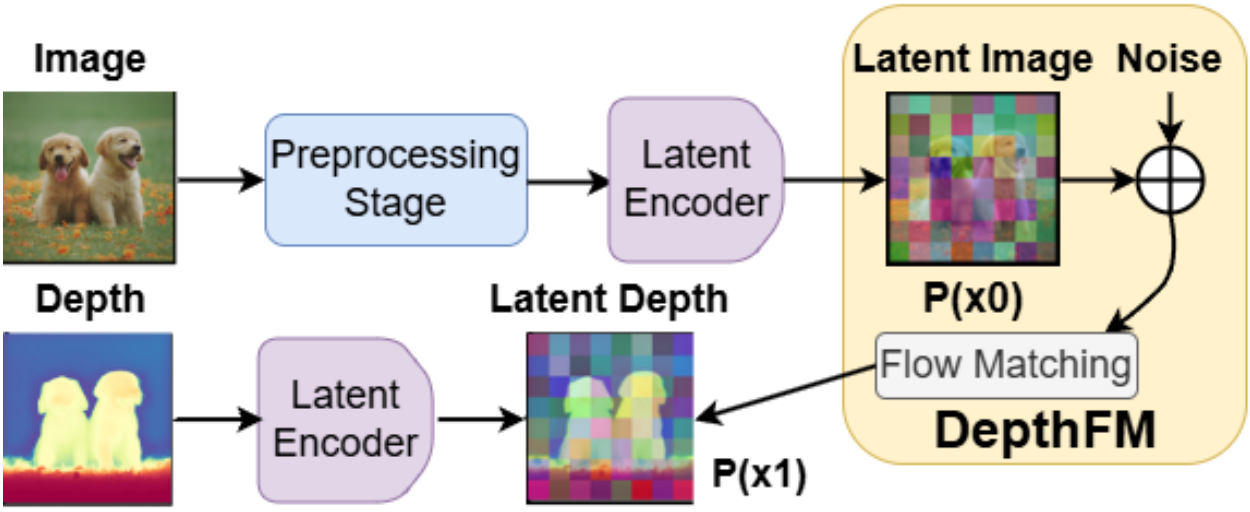

Figure 7: Overview of the Depth Prediction Model pipeline.

The basic idea of DepthFM is to use flow matching to regress the vector field between the image latent x_0 and the corresponding depth latent x_1. Detailed pipeline should be as follows:

1. **Flow Matching Objective**: In the latent space, a time-dependent vector field $u_t(x)$ is defined through an ordinary differential equation (ODE): $\frac{dx}{dt} = u_t(x)$, where x represents data points in the latent space, and $u_t(x)$ is the vector field at time t. The goal of flow matching is to learn a vector field $u_t(x|x_1)$ that transports the image latent representation x_0 to the depth latent representation x_1.
2. **Loss Function Definition**: The flow matching loss function is used to minimize the discrepancy between the predicted vector field and the true vector field: $L_{FM}(\theta) = \mathbb{E}_{t,x_0,x_1}\|v_t(x,t;\theta) - u_t(x|x_1)\|^2$, where $u_t(x|x_1)$ is the true vector field, and $v_t(x,t;\theta)$ is the vector field predicted by the model.

3. **Data Coupling**: To align the distributions between images and depth maps, Gaussian noise is applied to the image x_0, resulting in the following distribution: $x_t \sim \mathcal{N}(x_1 + (1-t)x_0, \sigma^2_{\text{min}})$. This ensures that the distributions of images and depth maps in latent space are effectively coupled.
4. **Training Process**: The image x_0 and the depth map x_1 are encoded into latent representations z_0 and z_1 using an encoder. The flow matching loss L_{FM} is used to train a vector field that maps z_0 to z_1. A decoder reconstructs the image and depth map from the latent space.
5. **Noise Augmentation Mechanism**: Gaussian noise is injected into the latent representations to smooth the distributions of image and depth map modalities, improving training stability and accelerating convergence.

3 Requirements & Verification

3.1 Robotic Arm

3.1.1 Camera

Requirement	Verification
1. The control subsystem could receive data from sensors and cameras as well as user's voice input correctly in real-time.	A. For safety concern, we firstly run our test cases of robotic control in software simulation.
2. The robotic arm works normally when data is entered.	B. The function of the robot arm is tested in the actual scenario.

3.1.2 Microphone

Requirement	Verification
1. The microphone can clearly capture user voice commands in real-time.	A. Voice input is tested in various noise conditions to ensure reliable audio capture.

3.1.3 STM32

Requirement	Verification
1. STM32 is able to receive, parse, and forward commands from the PC or ROS system with minimal delay.	A. Use logic analyzer or serial monitor to measure response time and confirm accurate data parsing.
2. STM32 shall periodically collect data from sensors (e.g., position, force, encoder) and transmit them accurately to the ROS system.	B. Inject known sensor data and confirm the integrity and timing of the data reported to ROS.
3.The microcontroller should handle communication errors, buffer overflows, and invalid packets gracefully without system crashes.	C. Introduce corrupted or delayed messages during testing to verify error handling mechanisms.

3.1.4 Motor

Requirement	Verification
1. Motors shall respond precisely to low-level PWM or control signals with minimal overshoot or delay.	A. Apply PWM signal tests at varying frequencies and loads to evaluate control fidelity and stability.
2. The direction, speed, and torque of motors must match the intended control outputs from the STM32.	B. Use encoder feedback and tachometers to compare target and actual motor behavior under different control signals.
3. Motors should operate within thermal, current, and voltage safety thresholds, and automatically shut down if thresholds are exceeded.	C. Monitor motor temperature, voltage, and current during runtime to ensure safety limits are not exceeded.
4.The motors shall maintain smooth and continuous motion during trajectory execution without jitter or stall.	D. Perform sudden changes in direction and load during trajectory execution to test for jitter, stall, or recovery failure.

3.1.5 End Effector

Requirement	Verification
1. The end effector can fully open and close to grasp objects within the rated 6 cm width.	A. The gripper is tested with objects of different sizes to verify opening and gripping performance.
2. The gripper can stably hold objects up to 200 g during movement.	B. The robotic arm is commanded to transport weighted test objects to confirm load handling and stability.

3.2 PC (ROS)

Requirement	Verification
1. Achieve low-latency response to commands from the STM32.	A. Measure the delay between receiving MQTT commands and initiating robotic arm movement. B. Run multiple command scenarios to verify consistent low-latency performance.
2. Ensure stable and reliable communication between the STM32 and the ROS system	C. Evaluate MQTT communication reliability under varying network conditions. D. Conduct stress testing to assess the system's robustness and accuracy in message handling.
3.Maintain high reliability and execution efficiency for tasks triggered by the STM32.	E. Confirm accuracy and timing consistency of task execution across different MQTT command sequences. F. Monitor system behavior over extended periods to validate long-term reliability and performance.

3.2.1 Communicator

Requirement	Verification
1. Ensure reliable and real-time communication between the PC and STM32 microcontroller.	A. Measure the communication delay and packet loss rate under different operational loads and network conditions. B. Conduct robustness testing by simulating packet drops or delays to verify system stability.
2. Handle message parsing and packaging accurately without data corruption.	C. Test with randomized payload data and edge-case messages to ensure correct encoding and decoding. D. Validate data integrity between sender and receiver.
3.Support bidirectional command and feedback flow.	E. Verify full-duplex communication by sending concurrent commands and feedback and checking for synchronization errors.

3.2.2 Path Translator

Requirement	Verification
1. Translate high-level path commands into low-level joint or actuator instructions accurately.	A. Compare the translated joint angles or control signals with expected values for known test paths. B. Test on real robot arm to ensure correct execution of generated instructions.
2. Ensure real-time performance for path computation.	C. Measure the computation time of the path translator under various path complexities to ensure responsiveness. D. Stress test the module with long or curved paths to observe latency and processing limits.
3.Maintain kinematic and motion constraints of the robotic arm during translation.	E. Validate that all generated paths respect joint limits, speed constraints, and avoid illegal positions. F. Simulate edge cases to ensure safe and predictable motion behavior.

3.3 AI Models

3.3.1 NLP Model

Requirement	Verification
1. Achieve accurate speech recognition and semantic understanding.	A. Design test cases with varied speech commands and noise conditions; record accuracy and compute average recognition rate. B. Test semantic label mapping accuracy under diverse natural language expressions.
2. Support real-time speech signal processing and labeling.	C. Measure time delay between receiving speech input and producing a label. D. Ensure the processing time does not exceed 200 ms.
3. Ensure adaptability to environmental noise and ambiguous inputs.	E. Evaluate performance of audio module in different environments (e.g., low noise, high noise, echo) and assess recognition stability. F. Assess semantic label module under various paraphrasing or ambiguous inputs for accuracy and robustness.

3.3.2 YOLOE (CV Model)

Requirement	Verification
1. Ensure images are correctly loaded without loss.	A. Perform a data stream test by saving images sent to both local PC and server, then manually compare the two to check for loss.
2. Ensure accurate object detection with clearly marked bounding boxes.	B. Conduct detection tests with varied backgrounds and object combinations, repeated at least 50 times to evaluate accuracy.
3. Ensure the target label (object name) is loaded correctly.	C. Log input labels during loading and compare them with expected values to verify correctness.
4. Select key frames from continuous camera input for model processing.	D. Use a frame counter to ensure that only one image is processed per instruction, validating resource-efficient frame selection.

3.3.3 Depth Prediction Model

Requirement	Verification
1. The model shall accurately convert monocular RGB images into depth maps using the DepthFM model.	A. Design test cases under varying depth scenarios and compare the output with ground truth for accuracy evaluation.
2. The model shall support real-time processing of monocular input images.	B. Measure processing latency and confirm it remains under 200 milliseconds to ensure real-time performance.
3. The model shall maintain accurate predictions under varying lighting conditions and partial visual occlusions.	C. Evaluate robustness by testing under different lighting environments and introducing visual noise or occlusions in the input images.

4 Tolerance Analysis

4.1 Robotic Arm

A key design consideration for the robotic arm lies in its ability to maintain stability, accuracy, and responsiveness under varying mechanical and command conditions. We focus our analysis on two main factors: gripping overload tolerance and trajectory execution tolerance.

Gripping Overload Analysis: The robotic arm's end effector is rated to handle objects up to 200 g. However, during testing, fluctuations in object weight or dynamic shifts during motion may introduce temporary overload conditions. We simulate gripping tasks with increasing weight loads and monitor motor current, claw slippage, and command feedback delay. When the load exceeds safe limits, the servo's internal protection triggers a stall mode to prevent mechanical damage. To address such situations, our system incorporates a threshold-based load monitoring mechanism. When stress readings approach the maximum safe torque, the STM32 microcontroller either adjusts the gripping force or aborts the task gracefully.

Trajectory Execution Analysis: Trajectory execution tolerance is primarily influenced by the timing precision of PWM signals and mechanical backlash. Under sudden command changes or during multi-axis coordinated movement, small errors may accumulate, resulting in end effector misalignment. To quantify this, we analyze trajectory deviation using camera feedback and inverse kinematics reconstruction. Our measurements show a maximum positioning error of $\leq 1^{\circ}$, consistent with the motor's specified backlash and return tolerance. This confirms that the mechanical and control system performs within expected tolerances.

4.2 PC(ROS)

Optimizing Gripping Strategies for Irregularly Shaped Objects: Irregularities in the shape of objects can pose significant challenges to the gripping accuracy of robotic arms. When handling complex or unevenly shaped items, the robotic arm may struggle to fully conform to the object's surface, resulting in deviations in the gripping position. To overcome this issue, it is essential to optimize the gripping strategy. Techniques such as multi-point gripping or the use of soft grippers can enhance the arm's ability to adapt to various shapes, thereby improving stability and accuracy during the gripping process.

4.3 Remote Server

NLP Model: The NLP module demonstrates a moderate level of tolerance to errors from the ASR subsystem. Given the potential presence of word-level transcription errors, the NLP model is designed to extract semantic labels using both pattern-based and semantic similarity methods. This dual approach allows the system to remain functional even when the input text is partially noisy or grammatically irregular. Furthermore, label extraction operates with a degree of redundancy by supporting multiple aliases or paraphrases of target objects, thereby improving robustness in real-world scenarios.

Computer Vision Model: The CV model module should effectively complete two tasks: identifying the objects against the background, and assigning a correct label to each object. Given that the input image is embedded to feature vectors, we can represent the model as an embedding $F(images) = features$. Then, we can use the loss function to optimize and evaluate the model. The first loss function refers to the model's ability of picking out objects from the background. In short, it can be represented as a binary cross-entropy loss. We first break the image into N grids. p_i^{obj} represent the probability of an object is in the ith grid, and $\hat{p}_i^{\text{obj}}$ is the corresponding ground truth with 1 of containing an object.

$$\mathcal{L}_{\text{obj}} = \frac{1}{N} \sum_{i=1}^{N} \text{BCE}(p_i^{\text{obj}}, \hat{p}_i^{\text{obj}})$$

Similarly, we can also represent the label assignment as another cross-entropy loss, where M is the number of objects, $p_{j,c}^{\text{cls}}$ is the probability of the object in region j be classified as label c, and $\hat{p}_{j,c}^{\text{cls}}$ as the ground truth.

$$\mathcal{L}_{\text{cls}} = \frac{1}{M} \sum_{j=1}^{M} \sum_{c=1}^{C} \text{BCE}(p_{j,c}^{\text{cls}}, \hat{p}_{j,c}^{\text{cls}})$$

These two loss function will be added together, and we may assign weights to them if we find the model perform much better in one loss than the other. Therefore, our final goal is to minimize the overall loss:

$$\mathcal{L} = \lambda_{\text{obj}} \mathcal{L}_{\text{obj}} + \lambda_{\text{cls}} \mathcal{L}_{\text{cls}}$$

Depth Prediction Model: The Depth Prediction module is inherently tolerant to moderate variations in image quality, such as motion blur, low light, or occlusion. DepthFM, as a learning-based model, can generalize across a range of visual conditions observed during training. In the event of partial frame corruption or incomplete data, preprocessing steps—such as temporal frame enhancement and noise reduction—help stabilize the input. Additionally, downstream components using depth information (e.g., grasp planning) incorporate fallback heuristics or confidence thresholds to handle uncertain or missing depth regions, further enhancing system-level robustness.

Data Transfer Analysis Between Computer and Server: Simulations have shown that data transfer delays between the computer and server can be limited to approximately 3-4 seconds. The primary factors affecting this latency include network speed, server processing capabilities, and the complexity of the data. To enhance processing efficiency, we can try to use Python libraries such as flash-attention, which significantly accelerate AI model computations. These libraries optimize real-time data analysis and decision-making processes, enabling faster and more efficient handling of information received from user devices. This approach not only reduces latency but also improves the overall responsiveness and accuracy of the system.

5 Cost Analysis

The cost consists of two parts. For labor part, our labor cost is calculated acccording to ECE industry average hourly wage, which is 15 dollars per hour. Starting from March to May, we spend two months, eight weeks on our senior design. Each week we spend 25 hours on project on average, so the total labor cost is 4 * 15 * 25 * 8 = 12000 USD.

For hardware part, our robotic arm, together with end effector, microphone and camera, the cost is 1379 RMB. So the total cost is 12000 * 7.3 + 1379 = 88979 RMB.

6 Schedule

Week	Tasks	Member
4/15	Finalize hardware setup: STM32, motors, camera, microphone. Connect end effector and test basic grasping.	Zixuan Zhang
	Pre-train and test NLP model on sample commands.	Junzhou Fang, Junsheng Huang
	Set up ROS environment on PC, confirm serial communication with STM32.	Zixin Zhu

Continued on next page

Table 2 – continued from previous page

Week	Tasks	Member
4/22	Integrate microphone with ASR pipeline (Wav2Vec2.0).	Junzhou Fang, Junsheng Huang
	Verify ROS control pipeline (PC → STM32 → motor).	Zixin Zhu, Zixuan Zhang
	Conduct mechanical stress test of arm + gripper.	Zixuan Zhang
4/29	Deploy YOLOE object detection model with ROS camera stream. Test YOLOE + NLP integration for object command matching.	Junzhou Fang, Junsheng Huang
	Refine ROS path planner and integrate with grasp commands.	Zixin Zhu
5/6	Conduct system-level integration (voice → vision → grasp). Test real-time instruction cycle with common items. Begin documentation of experimental results and failures.	All members
5/13	Demo dry run & presentation rehearsal.	All members
	Debug remaining issues with ROS control loop or grasp logic.	Zixin Zhu, Zixuan Zhang
	Finalize and polish report, diagrams, CAD drawings, and video.	Junzhou Fang, Junsheng Huang
5/19	Final demo and report submission.	All members

7 Ethics and Safety

Given that our target audience primarily includes the elderly and disabled, ethics and safety are crucial aspects of our design. This section is divided into two parts to comprehensively address these concerns.

7.1 Ethics

This work focuses on developing a robotic arm capable of picking up diverse small objects. Such a system has practical applications in home assistance, eldercare, logistics, and disaster recovery. By enabling reliable manipulation in unstructured environments, it contributes to building assistive and autonomous systems that improve safety, reduce human workload, and enhance quality of life. As ZJUI students, we are committed to upholding ethical standards and ensuring the integrity of our project. Our team will strictly

adhere to the IEEE Code of Ethics [7] and the ACM Code of Ethics [8]. We pledge to meet, but are not limited to, the following ethical responsibilities:

- Prioritize public safety, health, and well-being by adhering to ethical design principles and sustainable practices. Additionally, we are obligated to report any potential systemic risks that could lead to harm.
- Strive to benefit society by enhancing individual and collective understanding of both traditional and emerging technologies and their societal implications.
- Maintain honesty and integrity in all professional activities, strictly avoiding unethical conduct such as bribery or other illegal actions.

7.2 Safety

To ensure the safety of both team members and others, and to mitigate any potential hazards during the project, our team will strictly comply with the ECE 445 SAFETY GUIDELINES [9]. We will undertake, but are not limited to, the following safety measures:

- No team member is permitted to work alone in the laboratory at any time.
- All team members must complete mandatory safety training before being authorized to work in the laboratory.
- Any handling of battery charging or hazardous battery chemicals must be conducted in strict accordance with established safe usage guidelines.
- The robotic arm is designed to operate safely around humans, minimizing collision risks through motion planning and force-limited actuators. This ensures it can work in homes or care facilities without endangering users.
- The system incorporates grasp failure detection to avoid dropping or mishandling objects. This prevents accidental damage to the environment and ensures reliable object transfer.
- Adaptive force control mechanisms allow the arm to handle fragile objects without applying excessive pressure. This enhances safety when interacting with unknown or delicate items.
- Manual and automated emergency stop mechanisms are included to immediately halt motion in case of anomaly or user intervention. This provides a critical safety layer during deployment and testing.
- Safety-centric design builds user trust and promotes system adoption. By making robot actions predictable and explainable, users feel more comfortable interacting with the robot.

References

[1] B. Allen, C. Bagnati, E. Dow, *et al.*, "Report on the use of assistive robotics to aid persons with disabilities," *Undergraduate Coursework*, vol. 4, 2020. [Online]. Available: https://docs.lib.purdue.edu/ugcw/4/.

[2] P. Marti, M. Bacigalupo, and L. Giusti, "Assistive robots for the social management of health: A framework for robot design and human–robot interaction research," *International Journal of Social Robotics*, vol. 12, no. 4, pp. 681–702, 2020. DOI: 10.1007/s12369-020-00647-6. [Online]. Available: https://pmc.ncbi.nlm.nih.gov/articles/PMC7223628/.

[3] YahBoom, *Yahboom dofbot se robotic arm specifications*, Accessed: 2025-03-13, 2023. [Online]. Available: https://www.yahboom.com/tbdetails?id=562.

[4] A. Baevski, H. Zhou, A. Mohamed, and M. Auli, *Wav2vec 2.0: A framework for self-supervised learning of speech representations*, 2020. arXiv: 2006.11477 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2006.11477.

[5] A. Wang, L. Liu, H. Chen, Z. Lin, J. Han, and G. Ding, *Yoloe: Real-time seeing anything*, 2025. arXiv: 2503.07465 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2503.07465.

[6] M. Gui, J. Schusterbauer, U. Prestel, *et al.*, *Depthfm: Fast monocular depth estimation with flow matching*, 2024. arXiv: 2403.13788 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2403.13788.

[7] IEEE. ""IEEE Code of Ethics"." (2016), [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html (visited on 02/08/2020).

[8] Association for Computing Machinery, *ACM Code of Ethics and Professional Conduct*, Accessed: 2025-03-13, 2018. [Online]. Available: https://www.acm.org/code-of-ethics.

[9] University of Illinois Urbana-Champaign, *ECE 445 Safety Guidelines*, Accessed: 2025-03-13, 2025. [Online]. Available: https://courses.grainger.illinois.edu/ece445zjui/guidelines/safety.asp.