# Robot Writing for Sign Language

Ahilesh Rajaram, Anusha Manohar, Mukesh Javvaji, Yuxi Zhou
rajaram.a@northeastern.edu, manohar.an@northeastern.edu, javvaji.m@northeastern.edu, zhou.yuxi1@northeastern.edu

Link to the Github: https://github.com/anushamanohar/CS5335_Final_Project.git

Link to the google drive (for datasets):
https://drive.google.com/drive/folders/1M8HVJYrZNrS1S9WH45L3Uo6_VUvHdwca?usp=drive_link

## 1. Abstract

This paper presents the design and implementation of a novel assistive technology system that bridges the communication gap between sign language users and the written word. Our system combines computer vision and robotics to create an end-to-end solution that detects American Sign Language (ASL) gestures and physically transcribes them into text. Using a custom-trained YOLOv5 model, the system recognizes hand signs in real-time from webcam input with high accuracy, implementing a frame-stabilization technique to ensure reliable detection. The recognized signs are then converted to text and transmitted to a Pincher X 150 robotic arm, which physically writes the corresponding letters on a whiteboard using innovative kinematic solutions to overcome the limitations of the 4-DOF robotic platform. We address several technical challenges, including maintaining consistent pen-to-whiteboard contact through dynamic roll adjustment and parabolic path mapping, extracting and optimizing character stroke patterns from SVG font files, and ensuring smooth transitions between letters. Experimental results demonstrate the system's ability to successfully detect and transcribe multiple ASL alphabet signs with high reliability. This technology has potential applications in education, accessibility, and human-robot interaction domains.

## 2. Introduction

Sign language is a vital communication method for millions of individuals in the deaf and hard-of-hearing community worldwide. While sign language effectively facilitates communication among those who understand it, barriers persist when interacting with individuals unfamiliar with these visual-gestural languages. Traditional approaches to bridging this gap have focused on digital translations or virtual avatars, but these solutions often lack the tangible, permanent output that written communication provides.

The Robot Handwriting for Sign Language project addresses this challenge by creating a physical system that observes, interprets, and transcribes sign language into written text. By combining state-of-the-art computer vision techniques with precision robotics, our system serves as a mechanical interpreter that can transform ephemeral hand gestures into lasting written communication accessible to anyone who can read.

At the core of our system is a computer vision pipeline powered by a custom-trained YOLOv5 model that identifies American Sign Language (ASL) alphabet signs with high accuracy. The detected signs undergo a stabilization process to eliminate transient misdetections before being converted to text. This text is then processed by our robotic writing module, which transforms each character into a series of precise movement commands for a Pincher X 150 robotic arm equipped with a writing implement.

Implementing this system required overcoming several significant technical challenges. The 4-DOF limitations of the Pincher X 150 arm necessitated creative solutions to maintain consistent contact with a vertical writing surface. We developed mathematical models for dynamic roll adjustment and parabolic path mapping to ensure the pen remains perpendicular to the whiteboard throughout the writing process. Additionally, we created algorithms to extract and optimize character stroke patterns from SVG font files, ensuring natural-looking letter formation and smooth transitions between characters.

This paper details the design, implementation, and evaluation of our sign language to handwritten text system. We discuss the computer vision approach used for sign recognition, the mechanical and algorithmic solutions developed for robotic writing, the integration challenges encountered, and the performance of the complete system. We also explore potential applications in educational settings, accessibility tools, and human-robot interaction domains, along with directions for future enhancement and extension of this technology.

## 3. Methodology

### 3.1. Computer Vision Module

The Sign Language Robot project begins with the Computer Vision Module, which forms the foundation of the system. This component utilizes a custom-trained YOLOv5 model to detect American Sign Language gestures in real-time from webcam input. The module incorporates a sophisticated stabilization algorithm that requires five consecutive identical detections before confirming a sign, effectively eliminating transient misrecognitions. The user interface provides immediate visual feedback, supports both uppercase and lowercase modes, and offers essential text manipulation features including spaces, deletions, and clearing options.

### 3.2. SVG Path Processing Module

Working in tandem with the vision system, the SVG Path Processing Module serves as the bridge between detected signs and robot movements. This component extracts specific glyphs from SVG font files based on Unicode values and transforms complex vector paths into optimized waypoints suitable for robotic execution. The module employs adaptive sampling techniques that apply variable point density—creating finer resolution for curves while using fewer points for straight lines—resulting in smooth, natural-looking handwriting. Additionally, it organizes waypoints into distinct strokes and ensures consistent letter formation with standardized starting positions.

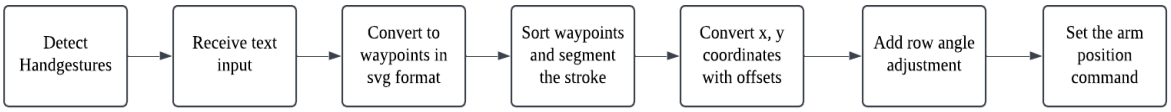
### 3.3. Robot Control Module

The Robot Control Module directly interfaces with the Pincher X 150 hardware through the InterbotixRobot SDK, translating processed waypoints into precise physical movements. To overcome the inherent limitations of the 4-DOF arm when writing on a vertical surface, this module implements two innovative solutions: a dynamic roll adjustment algorithm that keeps the pen perpendicular to the writing surface throughout execution, and a parabolic mapping function that strategically adjusts the x-coordinate based on the current y-position. These mathematical adaptations enable smooth, consistent pen contact despite the robot's kinematic constraints.

### 3.4. Integration Module

The Integration Module serves as the central coordinator, connecting all components into a cohesive system. It manages the complete pipeline from sign detection to physical writing, handling text processing, character spacing, and positioning logic for multi-character words. The module efficiently sequences pen-up and pen-down operations between strokes and characters, ensuring natural transitions and readable output. Supporting all these components, the Visualization Module provides essential debugging and verification capabilities, creating visual representations of planned paths and helping tune writing parameters for optimal performance.

### 3.5. Workflow and System Design




## 4. Requirements and Tools

### 4.1. Hardware Components

The primary hardware component of our system is the Pincher X 150 robotic arm, a 4-DOF manipulator selected for its affordability, accessibility, and sufficient precision for writing tasks. The arm features a gripper mechanism modified to securely hold a standard whiteboard marker. Additional hardware includes a standard webcam for video input, a stable mounting system for both the camera and robotic arm, and a vertically positioned whiteboard for writing output.

### 4.2. Software Environment

Our system was developed and deployed on Ubuntu 20.04 LTS as the operating system platform, providing a stable foundation for both the computer vision and robotics components. We utilized ROS Noetic (Robot Operating System) as the middleware framework, which facilitated communication between system modules and provided essential tools for robot control and visualization. Python 3 served as our primary programming language, allowing seamless integration with both computer vision libraries and robot control interfaces.

### 4.3. Development Tools and Libraries

For computer vision development, we leveraged several specialized tools:

- PyTorch and Ultralytics' YOLOv5 for model training and inference
- OpenCV for image acquisition and preprocessing
- CUDA for GPU acceleration during model training

The robotic control system utilized:

- Interbotix Python SDK for direct control of the Pincher X 150 arm
- svgpathtools for parsing and processing SVG font files
- Matplotlib for visualization and debugging of planned paths

### 4.4. Training Datasets

Our computer vision model was trained using two primary datasets:

1. ASL Alphabet dataset from Kaggle (https://www.kaggle.com/datasets/grassknoted/asl-alphabet), containing over 87,000 images of ASL hand signs
2. FER2013 dataset (https://www.kaggle.com/datasets/msambare/fer2013) was used to supplement training with additional hand positions and variations

These datasets were augmented with custom captures to improve robustness across different lighting conditions and users, resulting in a comprehensive training set that enabled reliable sign detection.

### 4.5. Integration Requirements

The complete system integration required careful calibration between the camera's field of view and the robot's workspace. This calibration process established the spatial relationship between detected signs and the writing

surface, ensuring that the complete pipeline from detection to physical writing maintained proper coordination. Additional requirements included consistent lighting conditions for optimal detection and a stable mounting arrangement to prevent calibration drift during operation.

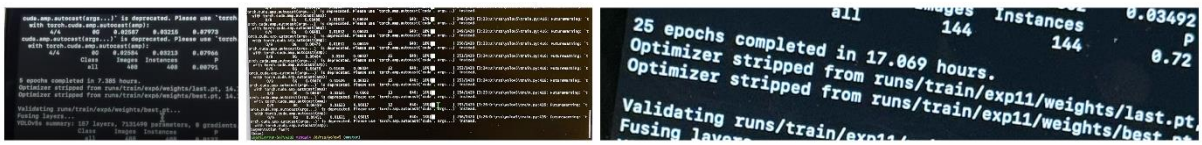
## 5. Computer Vision System

### 5.1. Model Architecture and Selection

The computer vision component of our Sign Language Robot utilizes a custom-trained YOLOv5 model for American Sign Language (ASL) detection. After extensive experimentation with multiple approaches including MediaPipe and various synthetic datasets, we determined that YOLOv5 offered the optimal balance of accuracy, processing speed, and deployment flexibility for our specific use case. The object detection architecture of YOLOv5 proved particularly effective for identifying hand positions and configurations in varied lighting conditions and backgrounds.

### 5.2. Dataset and Training

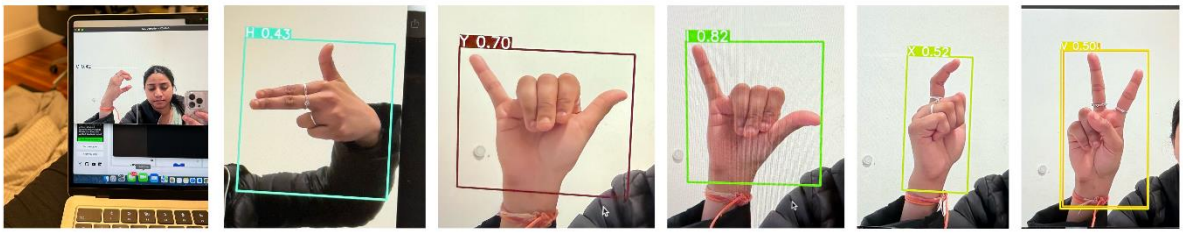
Our training approach involved curating a comprehensive ASL dataset from open repositories, which we then augmented with additional samples to improve robustness. The data preprocessing pipeline included normalization, augmentation (rotation, scaling, and brightness adjustments), and careful labeling to ensure consistent recognition across different users and environments. Training was conducted locally using GPU acceleration, with a configured virtual environment to maintain dependency consistency. The training process involved setting up appropriate file paths, configuring system adaptability settings, and implementing early stopping to prevent overfitting.




### 5.3. Detection Algorithm Implementation

The ASLDetector class in our system implements several sophisticated mechanisms to ensure reliable sign recognition:

1. Confidence Thresholding: We utilize a 0.25 confidence threshold to balance detection sensitivity with false positive reduction.
2. Prediction Stabilization: To overcome transient misdetections, we implemented a consecutive frame validation system that requires the same sign to be detected across five consecutive frames before confirming it as the user's intended gesture.
3. Real-time Visualization: The system provides immediate visual feedback by rendering bounding boxes around detected hand signs along with confidence scores, helping users adjust their signing for optimal recognition.
4. User Interface Controls: The detection module includes comprehensive on-screen instructions and keyboard controls for adding detected letters, managing text (backspace, clear), switching case modes, and saving frames for future analysis.



### 5.4. MediaPipe Experimentation

Before settling on YOLOv5, we explored MediaPipe for hand gesture recognition. This approach used holistic camera frames to extract hand landmarks and implemented an LSTM-based architecture for sequence classification. While we successfully trained models to recognize simple phrases like "thank you," "hello," and "I love you" using customized datasets stored in NPY format, the MediaPipe implementation showed significant limitations in angle invariance and person-independence. The same gestures achieved higher recognition accuracy only when placed at angles nearly identical to the training frames, making it less suitable for our general-purpose application.


### 5.5. Evaluation Metrics

The computer vision system was evaluated using four key metrics:

1. Accuracy: Measured by the model's ability to correctly identify ASL signs across different users and conditions.
2. Efficiency: Assessed through frame processing speed and resource utilization, ensuring real-time performance on standard hardware.
3. Consistency: Evaluated by testing recognition stability across multiple signings of the same letter and between different users.
4. Recognizability Range: Determined by the system's ability to maintain accurate detection across varied hand positions, angles, and distances from the camera.

The final implementation achieves high accuracy with minimal false positives, processes frames in real-time even on CPU-only systems, maintains consistent recognition across users, and accommodates a reasonable range of signing positions and styles.

## 6. Robot Writing System

### 6.1. Hardware and Framework Integration

The robotic writing component employs a Pincher X 150 robotic arm, a 4-DOF manipulator with precision control capabilities. Despite its limited degrees of freedom compared to industrial alternatives, we selected this platform for its accessibility, programmability, and sufficient reach for whiteboard writing applications. The system interfaces with the hardware through the InterbotixRobot SDK, providing abstracted control functions that translate planned waypoints into coordinated joint movements. This integration required careful calibration of the gripper system to maintain consistent marker grip pressure—tight enough to prevent slipping but gentle enough to avoid damaging the writing implement.

### 6.2. Dynamic Orientation Control

A significant challenge in implementing vertical writing with a 4-DOF arm was maintaining consistent contact between the marker tip and whiteboard surface throughout the writing motion. We developed two complementary solutions to address this limitation:

1. Dynamic Roll Adjustment: The system continuously modifies the roll angle of the end effector based on its current y-position using a carefully calibrated formula (roll = np.round(-4.26*-(stroke[i][0]+y_offset), 3)). This adjustment ensures the marker remains perpendicular to the writing surface regardless of position, preventing uneven pressure that would result in inconsistent line quality.
2. Parabolic X-Axis Mapping: We implemented a mathematical transformation function (map_y_to_x) that dynamically adjusts the x-coordinate based on the current y-position following a parabolic curve. This compensation accounts for the arm's kinematic constraints when approaching different areas of the whiteboard, maintaining optimal contact without requiring additional degrees of freedom.

### 6.3. Path Generation and Optimization

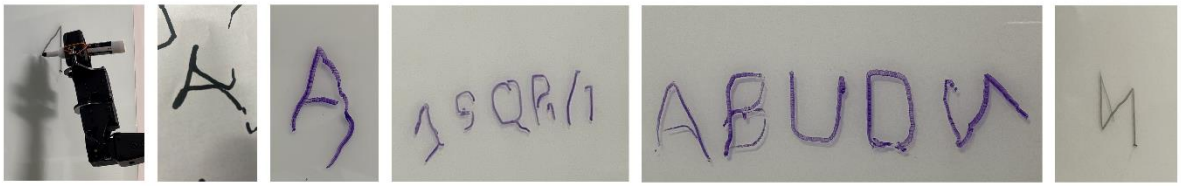
The ReadSVG class handles the complex process of transforming font glyphs into executable robot paths:

1. SVG Parsing and Extraction: The system extracts specific character glyphs from standard SVG font files using their Unicode values, accessing the vector path data that defines each letter's structure.
2. Adaptive Waypoint Generation: Rather than using uniform sampling, our algorithm implements adaptive density control that places more waypoints along curves and fewer along straight segments. This optimization produces more natural-looking writing while minimizing unnecessary robot movements.
3. Stroke Organization: The system identifies and organizes waypoints into logical strokes, properly sequencing pen-up and pen-down operations to maintain natural writing flow. We implemented specialized sorting algorithms to ensure each letter's starting point remains consistent, creating predictable and readable output.

### 6.4. Writing Coordination and Execution

The Writer class orchestrates the physical writing process through a series of carefully sequenced operations:

1. Path Execution: Each letter's waypoints are processed sequentially, with the arm performing precise transitions between points while maintaining appropriate velocity profiles for smooth writing.
2. Pen Management: The system implements controlled pen-up and pen-down movements between strokes, lifting the marker just enough to prevent unwanted marks while minimizing travel time.
3. Position Tracking: Throughout execution, the robot tracks its last position to ensure continuous writing across multiple letters, using y-offset calculations to maintain proper spacing and alignment.
4. Padding and Layout Management: A specialized padding function ensures proper spacing between letters, particularly important for maintaining readability when writing multiple characters in sequence.



### 6.5. Evaluation Metrics

The robotic writing system was evaluated using several key performance metrics:

1. Legibility: We assessed the readability of the produced text through visual inspection and user testing,

ensuring that the written characters were clearly identifiable to readers unfamiliar with the system.

2. Consistency: Measured by comparing multiple instances of the same character written by the robot, analyzing variation in stroke width, character dimensions, and overall appearance.
3. Completion Rate: Evaluated by the percentage of successfully completed characters without failures such as marker slippage or missed contact with the writing surface.
4. Writing Speed: Quantified as the time required to complete full characters and words, balanced against quality considerations to optimize for both speed and legibility.
5. Resource Efficiency: Analyzed by monitoring power consumption, computational demands, and mechanical wear during extended writing sessions.

After extensive testing and refinement, the robot writing system achieves reliable, consistent letter formation with natural appearance, effectively translating the detected ASL signs into readable text on the whiteboard.

## 7. System Setup and Implementation

### 7.1 System and Robotic Arm Simulation Setup

Our development process began with establishing a robust environment for both simulation and physical testing. This initial phase encountered several technical challenges that required systematic troubleshooting. A significant hurdle emerged with the package management system, producing the error "E: Unable to parse package file /var/lib/apt/lists/partial/archive.ubuntu.com_ubuntu_dists_precise_multiverse_i18n_Index (1)." This stemmed from network connectivity issues and virtual machine configurations. After investigation, we addressed these issues by migrating from VirtualBox to VMware and implementing Ubuntu mirror repositories to ensure stable package access.

The development environment setup required careful management of dependencies and package sources. We resolved duplicate source problems by clearing the local package cache, updating repository lists, and transitioning to a currently supported Ubuntu distribution. This provided a stable foundation for both the computer vision processing and robot control components.

Hardware integration presented another layer of complexity. Initially, the USB device connecting to the Pincher X 150 robotic arm wasn't recognized properly by the system. We systematically adjusted USB port configurations on both the host Windows system and the virtual machine, verifying proper enumeration and permissions to restore device communication. This careful configuration ensured reliable control signals could be transmitted to the robotic arm without latency issues that would compromise writing precision.

### 7.2. Robotic Arm Testing and Calibration

Once the basic system was operational, we conducted extensive testing of the robotic arm's capabilities and limitations. This involved systematically mapping the arm's range of motion within its workspace, particularly focusing on the areas relevant to whiteboard writing. We identified several "dead angles" where the arm's kinematic constraints prevented reliable positioning, allowing us to define usable boundaries for our writing area.

During the testing phase, we created a keyboard-controlled interface to manually position the arm, which proved invaluable for discovering the practical two-dimensional coordinate range for writing operations. This hands-on approach allowed us to match the origin points between the simulation environment and the physical arm, ensuring consistency between planned and executed movements. We deliberately tested boundary conditions and invalid inputs to understand the system's failure modes and implement appropriate safeguards.

### 7.3. Implementation and Execution

The implementation process followed an iterative development approach. For the computer vision component, we began with basic ASL letter detection and progressively refined the model through training iterations. The YOLOv5 model was initially trained on a general dataset, then fine-tuned with additional samples focused on challenging recognition cases identified during testing.

For the robotic writing system, we started with basic letter formation and gradually incorporated the dynamic adaptations needed for consistent writing. The SVG path extraction and transformation pipeline underwent multiple refinements to improve letter quality and natural appearance. Each iteration addressed specific issues observed during testing, such as inconsistent pen pressure or unnatural stroke execution.

System integration occurred incrementally, first connecting basic letter detection to simple robotic movements, then adding the advanced features such as text buffering, case switching, and multi-character writing. At each stage, we conducted comprehensive testing with different users and lighting conditions to identify and address edge cases.

The final system execution follows a structured pipeline: The webcam captures frames that are processed through the YOLOv5 model, detected signs undergo stability verification, confirmed letters are converted to text, this text is transformed into waypoints through SVG parsing, and finally, these waypoints guide the robotic arm's movements to create physical writing on the whiteboard. This complete pipeline functions as an integrated system that successfully translates hand gestures into legible written text.

## 8. Limitations and Challenges

### 8.1. Computer Vision System

The vision system encountered several performance-limiting factors during development and testing. These challenges highlighted both hardware constraints and the complexities of creating a robust sign language detection system. From our research process, we face challenges in reaching accurate gesture detection. For static images, dark environment with blurred contours creates challenges for training the Yolov5 model. When recognizing the gesture letters in the bounding box, the shadow between overlapping fingers creates confusion for the model to distinguish similar gestures such as “M” and “T”.

Key limitations included:

- Sensitivity to ambient lighting conditions, with degraded performance in low-light environments
- Difficulty distinguishing signs against distant areas or moving backgrounds
- Reduced accuracy when encountering signing styles that differed significantly from the training dataset
- Limited ability to recognize dynamic signs involving motion

### 8.2. Robotic Writing System

The robotic writing component faced physical and mechanical challenges that impacted writing quality and system usability. These limitations stemmed primarily from the inherent constraints of the hardware platform.

Major challenges included:

- The Pincher X 150's 4-DOF limitation causing inconsistent pressure at whiteboard edges
- Gradual drift in precision during extended operation requiring recalibration
- Necessarily slow writing speed to maintain quality and consistent marker contact
- Variations in performance with different marker types and ink levels
- Limited usable writing area due to the robot's reach constraints

### 8.3. Integration Challenges

Combining the vision and robotic systems created additional complexities beyond those faced by each component individually. These integration issues affected the overall user experience and deployment flexibility.
Notable integration challenges:

- Timing synchronization between sign detection and writing execution
- Limited portability due to calibration requirements and physical setup needs
- High computational demands from simultaneous vision processing and robot control
- User confusion from variable delays between signing and completed writing
- Resource constraints affecting deployment options and system cost

## 9. Future Work

The Sign Language Robot project demonstrates promising capabilities, but several avenues for improvement and expansion remain. Future development could focus on enhancing the vision system with more diverse training data to improve recognition across different users and environments. Implementing deep learning models capable of recognizing dynamic signs and full ASL phrases would significantly expand communication capabilities. From existing research, there are approaches in applying WiFi CSI with vision-based frameworks like YOLOv8 and Mediapipe. [1] From this source, we can refine the gesture recognition pipeline targeting the inaccuracy caused by depth calculation. Hardware improvements could include upgrading to a robotic arm with additional degrees of freedom to improve writing quality and speed. Miniaturization of the system components would enhance portability and deployment flexibility. Integration with text-to-speech capabilities could create a bidirectional communication bridge between signed and spoken language, further expanding accessibility applications.
Key areas for future development:

- Expanded training dataset encompassing greater user diversity and consecutive frames
- Explore hybrid pipeline in merging yolov5 and mediapipe
- Hardware upgrades for improved writing precision and speed
- Miniaturization for enhanced portability
- Integration with complementary accessibility technologies

## 10. Conclusion

The Robot Handwriting for Sign Language project successfully demonstrates the feasibility of creating a physical bridge between American Sign Language and written text. By combining computer vision and robotic control technologies, we've developed a system that can observe hand gestures, interpret them as letters, and physically transcribe them onto a writing surface. Despite the challenges inherent in both the detection and writing components, our approach effectively translates ephemeral visual communication into permanent written form.
The implementations of the YOLOv5-based detection system and the innovative solutions for robotic writing demonstrate how creative engineering can overcome significant technical constraints. Our dynamic roll adjustment and parabolic mapping functions allowed a limited-DOF robot to perform writing tasks typically requiring more sophisticated hardware. These innovations showcase the potential for accessible robotics to address meaningful communication challenges.
While current limitations restrict practical deployment in some scenarios, the foundation established by this project provides a clear pathway for future enhancements. The system not only serves as a potential assistive technology but also demonstrates valuable approaches for human-robot interaction and computer vision applications. By continuing to refine and expand this technology, we can further bridge communication gaps and create more inclusive environments for the deaf and hard-of-hearing community.

References

1. Boudlal, Hicham, Mohammed Serrhini, and Ahmed Tahiri. “A Novel Approach for Simultaneous Human Activity Recognition and Pose Estimation via Skeleton-Based Leveraging WIFI CSI with Yolov8 and Mediapipe Frameworks.” Signal, Image and Video Processing 18, no. 4 (February 29, 2024): 3673–89. https://doi.org/10.1007/s11760-024-03031-5.