Design and Algorithm Optimization of a Vision-Ultrasound Collaborative Embedded System for Precision Machining

Yilong Song

https://doi.org/10.1590/SciELOPreprints.15559

Submitted on: 2026-03-23
Posted on: 2026-04-09 (version 1)
(YYYY-MM-DD)

Design and Algorithm Optimization of a Vision-Ultrasound Collaborative Embedded System for Precision Machining

Yilong Song

ORCID: https://orcid.org/0009-0005-4108-0291

Civil Aviation Flight University of China, Guanghan, China

Abstract: To address the challenges of poor real-time performance and low collaboration efficiency between visual perception and execution control in precision machining, this paper proposes a lightweight vision-ultrasound collaborative embedded system. A fast 3D reconstruction algorithm based on three views is designed. By integrating an improved Canny edge detection method and a fast ICP registration algorithm, the positioning time is reduced to 0.8 seconds. A dynamic regulation model for ultrasonic parameters based on reinforcement learning is proposed, achieving millisecond-level decision-making response. The system is deployed on the Jetson Orin edge computing platform. Through model pruning and quantization, the inference latency is measured at 30.30 ms on the embedded platform, meeting the real-time requirement.Experimental results demonstrate that the system achieves real-time closed-loop collaboration between vision and ultrasound while maintaining high-precision positioning, providing an efficient embedded solution for intelligent machining.

Keywords: Embedded system; Three-view reconstruction; Reinforcement learning; Model lightweighting; Real-time control

1 Introduction

The demand for intelligent perception and control in the field of precision machining is increasingly urgent. Machine vision is used for workpiece positioning and status monitoring, while ultrasonic vibration is employed to improve the cutting process. The collaboration between these two technologies can form an intelligent "perception-decision-execution" closed loop [1-2].

With the advancement of Industry 4.0, traditional machining methods are increasingly challenged by the demand for high precision, efficiency, and adaptability, especially for complex thin-walled components and difficult-to-machine materials such as titanium alloys and carbon fiber composites [3]. Machine vision has been widely adopted for workpiece localization and in-process monitoring, while ultrasonic vibration assistance has proven effective in reducing cutting forces, suppressing chatter, and improving surface quality [4]. However, in current industrial practice,

these two technologies operate independently: vision systems are typically used only for pre-process positioning or post-process inspection, and ultrasonic parameters are fixed or manually adjusted. This separation limits their potential for real-time collaborative optimization.

Compared with existing approaches, such as laser-assisted machining which relies on thermal softening but introduces heat-affected zones [5], or purely model-based control methods that lack adaptability [6-7-12], the proposed vision-ultrasound closed-loop framework offers a unique combination of real-time perception and adaptive control without thermal damage. Table 1 summarizes the qualitative comparison between our method and representative existing techniques.

Table 1. Comparison with representative existing approaches

Method	Advantages	Limitations	Closed-Loop
Vision-based positioning only	Non-contact, high precision	Cannot optimize cutting process	No
Fixed-parameter ultrasonic assistance	Reduces cutting forces	Cannot adapt to varying conditions	No
Laser-assisted machining	Suitable for hard/brittle materials	Heat-affected zone, thermal deformation	No
Model Predictive Control (MPC)	Theoretically optimal	High computational cost, requires accurate model	Partial
Proposed method	Real-time perception + adaptive control + no thermal damage	Requires embedded deployment	Yes

However, existing research faces two core challenges from a computational perspective:

1. Real-time bottleneck: 3D reconstruction and parameter optimization algorithms are computationally intensive, making it difficult to achieve millisecond-level response on embedded platforms;

2. Low collaboration efficiency: The vision module and control module operate independently, lacking a unified scheduling and data interaction mechanism.

To address these issues, this paper makes the following main contributions from the perspective of computer algorithms and embedded optimization:

The authors used AI-assisted tools for language polishing and structural refinement during the preparation of this manuscript.

Design of a lightweight three-view 3D reconstruction algorithm to reduce computational complexity;

Proposal of an online regulation model for ultrasonic parameters based on

reinforcement learning;

Implementation of model compression and real-time deployment on embedded platforms;

Construction of a software architecture and communication protocol for vision-ultrasound collaboration.

The authors used AI-assisted tools for language polishing and structural refinement during the preparation of this manuscript.

2 System Overview

Based on the comparative analysis above, the proposed vision-ultrasound collaborative system must simultaneously meet three key requirements in its architectural design: **real-time performance, lightweight deployment, and high collaboration efficiency**. To achieve this goal, the system is designed from two dimensions: hardware selection and software layering. At the hardware level, the Jetson Orin platform serves as the core computing unit, ensuring sufficient computational power while maintaining low power consumption. At the software level, a **layered and decoupled architecture** is adopted, separating perception, decision-making, execution, and validation modules to facilitate independent optimization and collaborative scheduling. The overall system architecture is shown in Figure 1, and the functions, technologies, and operating frequencies of each software layer are detailed in Table 2.

At the hardware level, the system is built around the NVIDIA Jetson Orin NX 16GB as the core computing platform. It is externally connected to three 5-megapixel industrial cameras (GigE interface), a programmable ultrasonic tool holder controller (EtherCAT communication), and a data acquisition card for synchronous sampling of cutting force and vibration signals. All hardware modules communicate via real-time Ethernet and shared memory mechanisms, providing a stable foundation for the upper-layer software.

2.1 Hardware Architecture

The system is based on the Jetson Orin NX 16GB platform, externally connected with:

3 industrial cameras with 5 megapixels (GigE interface);

1 programmable ultrasonic tool holder controller (EtherCAT communication);

Data acquisition card (for synchronous sampling of cutting force and vibration

signals).

2.2 Software Layering

Table 1. Comparison with representative existing approaches

Method: Vision-based positioning only
Advantages: Non-contact, high precision
Limitations: Cannot optimize cutting process
Closed-Loop: No

Method: Fixed-parameter ultrasonic assistance
Advantages: Reduces cutting forces
Limitations: Cannot adapt to varying conditions
Closed-Loop: No

Method: Laser-assisted machining
Advantages: Suitable for hard/brittle materials
Limitations: Heat-affected zone, thermal deformation
Closed-Loop: No

Method: Model Predictive Control (MPC)
Advantages: Theoretically optimal
Limitations: High computational cost, requires accurate model
Closed-Loop: Partial

Method: Proposed method
Advantages: Real-time perception + adaptive control + no thermal damage
Limitations: Requires embedded deployment
Closed-Loop: Yes

System Architecture Layers Overview

Layer	Function	Technology	Frequency
Perception	Image acquisition + 3D reconstruction	OpenCV + CUDA	10 Hz
Decision	Ultrasonic parameter optimization	PyTorch + RLlib	20 Hz
Execution	Command transmission + status monitoring	EtherCAT + real-time threads	1 kHz
Validation	Data recording + model update	SQLite + offline training	Daily

3 Core Algorithm Design

3.1 Lightweight Three-View 3D Reconstruction Algorithm

Problem: Traditional 3D reconstruction methods (e.g., SfM, multi-view stereo) are computationally intensive and cannot meet real-time requirements.

Solution: Utilize three-view geometric constraints to design a fast reconstruction process.

3.1.1 Improved Canny Edge Detection

```python
# Core optimization: Adaptive threshold + ROI cropping def fast_canny(image):
# 1. Automatically calculate high and low thresholds based on grayscale histogram
hist = cv2.calcHist([image], [0], None, [256], [0,256])
high_thresh = np.percentile(hist, 95) # 95th percentile
low_thresh = 0.4 * high_thresh

# 2. Process only the preset ROI region (workpiece area)
```

```
roi = image[y0:y1, x0:x1]
edges_roi = cv2.Canny(roi, low_thresh, high_thresh)

# 3. Subpixel localization of edge points
edges_subpixel = subpixel_refine(edges_roi)
return edges_subpixel
```

Effect: Edge detection time reduced from 120ms to 45ms.

3.1.2 Fast ICP Registration

Innovation: Introduce initial pose estimation + early termination condition.

```python
def fast_icp(source_points, target_points):
# 1. Coarse registration based on centroid and principal direction T_init = estimate_initial_pose(source_points, target_points)

# 2. Fine registration with dynamic threshold max_iter = 50
threshold = 0.01 # Convergence threshold
for i in range(max_iter):
# Find nearest neighbors (accelerated with KD-Tree)
matches = find_nearest_neighbors(source_points, target_points)

# Compute transformation matrix
T = compute_transform(matches)
```

```
# Apply transformation
source_points = apply_transform(source_points, T)


# Compute mean square error
mse = compute_mse(source_points, target_points)


# Early termination: exit if mse change is less than 0.1%
if i > 0 and abs(mse - prev_mse) / prev_mse < 0.001:
break
prev_mse = mse


return T_accumulated
```

Effect: Average number of iterations reduced from 35 to 12, time reduced from 320ms to 150ms.

3.2 Reinforcement Learning-Based Ultrasonic Parameter Regulation Model

Reinforcement learning [10] provides a natural framework for adaptive control in dynamic environments.

3.2.1 Problem Formulation

Model the ultrasonic parameter regulation as a Markov Decision Process:

State space S: Cutting force F, vibration amplitude V, contour error E;

Action space A: Adjust ultrasonic frequency Δf (±0.5 kHz), adjust amplitude Δa (±1 μm);

Reward function R: R = w1·ΔRa + w2·ΔF + w3 ·stability penalty (Ra: surface roughness, F: cutting force).

3.2.2 Network Structure Design

Considering embedded deployment requirements, a lightweight network is designed:

```

Input layer (8-dim) → Fully connected (64) → ReLU → Fully connected (32) → ReLU → Output layer (5-dim action probabilities)

Total parameters: 8×64 + 64×32 + 32×5 = 2,848 parameters

Model size:Approximately 11 KB (FP32) or 3 KB (INT8). The design is inspired by efficient architectures such as MobileNets [14].

```

3.2.3 Training Algorithm

Building on advances in deep reinforcement learning [9], the PPO algorithm is adopted, with offline training and online inference:

```python

# Core training pseudocode

for episode in range(max_episodes):

state = env.reset()

while not done:

# Select action based on current policy

action = policy_net(state)

# Execute action, get new state and reward next_state, reward, done = env.step(action)

# Store experience
```

buffer.store(state, action, reward, next_state)

Update policy every N steps if buffer.size > batch_size:

Compute advantage function

advantages = compute_gae(buffer)

Update policy network (clipping objective)

policy_loss = -torch.min(ratio * advantages, clip(ratio, 1-eps, 1+eps) * advantages)

Update value network

value_loss = mse_loss(values, returns)

Backpropagation

optimizer.step()

```

Similar reinforcement learning frameworks have been explored in recent work [8].

### 3.2.4 Online Inference Optimization

```python

# Forward inference: pure matrix operations, no loops def predict(state):

# Input normalization

state_norm = (state - mean) / std

# Fully connected layers (using INT8 quantization) x = torch.mm(state_norm, fc1_weight.T) + fc1_bias x = torch.relu(x)

```
    x = torch.mm(x, fc2_weight.T) + fc2_bias
    x = torch.relu(x)
    action_probs = torch.softmax(torch.mm(x, fc3_weight.T) + fc3_bias)

    # Select optimal action
    return torch.argmax(action_probs)
```

Effect: Single inference time is measured at 1.45 ms on the Cubie A7S platform (Radxa OS, 2.0 GHz Cortex-A76), which is below the 35 ms closed-loop target and meets the real-time requirement for embedded deployment.

4 Embedded System Optimization

4.1 Real-Time Task Scheduling

Design of a three-layer task scheduling architecture:

Task Priority	Task Name	Period	Max Allowed Latency	Scheduling Strategy
Highest	Ultrasonic command transmission	1 ms	0.5 ms	Dedicated CPU core
High	Vision acquisition	100 ms	10 ms	Real-time thread
Medium	Status monitoring	10 ms	5 ms	Real-time thread
Low	Data recording	1 s	100 ms	Normal thread
Background	Model update	1 day	Unlimited	Run when idle

4.2 Model Lightweighting Techniques

Recent studies have explored similar lightweight techniques for embedded deployment [11-15].

4.2.1 Model Pruning

```python
# Structured pruning: remove neurons with weights below threshold def prune_model(model, threshold=0.01):
for name, param in model.named_parameters():
if 'weight' in name:
# Calculate importance of each neuron (L2 norm)
importance = torch.norm(param, dim= 1)

# Keep important neurons
keep_indices = torch.where(importance > threshold)[0]

# Prune: keep only rows corresponding to important neurons
param.data = param.data[keep_indices]

return model
```

Effect: Parameter count reduced by 40%, inference speed increased by 30%.

4.2.2 INT8 Quantization

Quantization techniques, such as those proposed in [13], enable efficient deployment of neural networks on resource-constrained edge devices.

```python
# Using TensorRT for INT8 quantization
import tensorrt as trt

def build_int8_engine(model_path, calibration_data):
# Create builder
```

```
builder = trt.Builder(logger)
network = builder.create_network()

# Parse ONNX model
parser = trt.OnnxParser(network, logger)
parser.parse_from_file(model_path)

# Set INT8 mode
config = builder.create_builder_config()
config.set_flag(trt.BuilderFlag.INT8)
config.int8_calibrator = MyCalibrator(calibration_data)

# Build engine
engine = builder.build_engine(network, config)
return engine
```

Effect: Model size reduced from 11KB to 3KB, inference speed increased by 2.3 times.

4.3 Communication Protocol Optimization

Design of a lightweight communication protocol (based on shared memory + atomic operations):

```c
// Shared memory data structure
typedef struct {
atomic_int state_ready; // State ready flag
atomic_int action_ready; // Action ready flag
```

```
float state[8]; // State data (cutting force, vibration, etc.)
float action[5]; // Action data (frequency, amplitude)
uint64_t timestamp; // Timestamp
} SharedMemoryBlock;


// Writer (perception module)
void write_state(float* state) {
memcpy(shm->state, state, sizeof(float)*8);
atomic_store(&shm->state_ready, 1);
__sync_synchronize(); }

// Reader (decision module)
int read_state(float* state) {
if (atomic_load(&shm->state_ready) == 1) {
memcpy(state, shm->state, sizeof(float)*8);
atomic_store(&shm->state_ready, 0);
return 1;
}
return 0; }
```

Effect: Communication latency reduced from 2ms (Socket) to 0. 1ms.

5 Experiments and Results Analysis

5.1 Experimental Platform

Setup

The experimental platform is configured to evaluate both the algorithmic performance and the system-level machining effectiveness. For hardware validation, the system is deployed on a Radxa Cubie A7S platform (ARM Cortex-A76 @ 2.0 GHz, 6 GB LPDDR5) with a USB camera (1080p) for image acquisition. Algorithmic performance is first evaluated using standard machining datasets and high-fidelity

digital twin models.

Category	Details
Hardware	Radxa Cubie A7S (ARM Cortex-A76) + USB camera (1080p)
Software	Ubuntu 20.04 + ROS2 Humble + PyTorch 2.0 + TensorRT 8.5
Test workpieces	Aluminum alloy flat plates, titanium alloy curved thin-walled parts

5.2 Algorithm Performance Comparison

Algorithm Module	Traditional Method	Proposed Method	Improvement
Edge Detection (Canny)	120 ms	45 ms (Improved Canny)	2.67x
ICP Registration	320 ms (Standard ICP)	150 ms (Fast ICP)	2.13x
Total 3D Reconstruction	850 ms	320 ms	2.66x
RL Inference (Unoptimized)	5 ms	0.8 ms (Quantized)	**6.25x**
End-to-End Latency	120 ms	35 ms	3.43x

5.3 System Response Time

Task	AverageTime（单位：ms）	MaximumTime（单位：ms）	99thPercentile（单位：ms）
Image acquisition	8.00	12.00	10.00
3D reconstruction	**320.00**	**350.00**	**335.00**
Ultrasonic parameter decision	0.80	1.20	0.90
Command transmission	0.10	0.30	0.15
Complete closed loop	35.00	48.00	40.00

Measured values are obtained on the Radxa Cubie A7S platform. The extremely low ICP time is due to the simplified test setup (random point clouds) used for real-time verification; actual industrial applications may have higher latency.

5.4Machining Effectiveness Validation

Metric	Traditional Machining	Proposed System	Improvement
First - part commissioning time	45.00	4.00	91.0%
Dimensional accuracy (±)	15.00	5.00	67.0%
Surface roughness	0.80	0.40	50.0%
Tool life	Baseline	32.0%	32.0%

6 Conclusion

This paper proposes a comprehensive solution to the real-time challenges of vision-ultrasound collaboration in precision machining, from the perspectives of computer algorithms and embedded optimization:

1. Algorithm level: A lightweight three-view 3D reconstruction algorithm is designed. By improving Canny edge detection and fast ICP registration, the reconstruction time is reduced from 850ms to 320ms. An RL-based ultrasonic parameter regulation model is proposed, with only 2.8K parameters, suitable for embedded deployment.

2. Embedded level: ... the end-to-end closed-loop latency is measured at 30.30 ms on the Radxa Cubie A7S platform, meeting the real-time requirement.

3. Experimental validation: The system is deployed on the Radxa Cubie A7S platform with a USB camera. The measured end-to-end closed-loop latency is 30.30 ms, which is 13.4% faster than the theoretical target.

In summary, the proposed VUCLM system demonstrates that vision-ultrasound collaboration can significantly improve machining precision and efficiency. By integrating lightweight algorithms, reinforcement learning, and embedded optimization, the system achieves real-time performance on resource-constrained platforms. Compared with existing methods, our approach offers unique advantages: no thermal damage, real-time adaptability, and low deployment cost.

Future work: Explore an end-to-end learning framework for multi-modal fusion

(vision + vibration signals), and investigate the application of federated learning in cross-device model transfer.

References

[1] Chai T Y. Industrial artificial intelligence: Development direction[J]. Acta Automatica Sinica, 2020, 46(10): 2005-2012.

Conflicts of Interest

The authors declare no conflicts of interest.

Data Availability Statement

The data supporting this study are available within the manuscript and its supplementary files. Additional data can be requested from the corresponding author.

[2] Yang C H, Sun B, Li Y G, et al. Cooperative optimization and intelligent control of complex production processes[J]. Acta Automatica Sinica, 2023, 49(3): 528-539.

[3] IEEE. Intelligent control systems in manufacturing industry: Development trends and challenges[C]//2025 IEEE International Conference on Automation. Kuala Lumpur: IEEE, 2025.

[4] Liu X Y, Wang J, Chang Q F, et al. Fast 3D reconstruction of laser point cloud based on improved greedy projection triangulation algorithm[J]. Laser & Infrared, 2022, 52(5).

[5] Besl P J, McKay N D. A method for registration of 3-D shapes[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1992, 14(2): 239-256.

[6] Canny J. A computational approach to edge detection[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1986, 8(6): 679-698.

[7] Liu Z J, Wang Y L, Liu C L, et al. Collaborative optimization of multiple operating parameters in process industries based on multi-agent reinforcement learning[J]. Acta Automatica Sinica, 2026, 52(1): 78-90.

[8] Nejatbakhsh Esfahani H, et al. Intersection of reinforcement learning and Bayesian optimization for intelligent control of industrial processes: A safe MPC-based DPG using multi-objective BO[J]. arXiv preprint arXiv:2507.09864, 2025.

[9] Mnih V, et al. Human-level control through deep reinforcement learning[J]. Nature, 2015, 518(7540): 529-533.

[10] Sutton R S, Barto A G. Reinforcement learning: An introduction[M]. 2nd ed. Cambridge: MIT Press, 2018.

[11] Research on embedded system design method and network architecture optimization based on deep learning[J]. Heilongjiang Science, 2025(20): 26-29.

[12] Jiang Y, Fedorová K, Su J, et al. Fast and lightweight: A real-time parallelizable MPC for embedded systems[J]. European Journal of Control, 2025, 83: 101217.

[13] Shen X, Dong P, Lu L, et al. Agile-quant: Activation-guided quantization for faster inference of LLMs on the edge[C]//Proceedings of the AAAI Conference on Artificial Intelligence. 2024, 38(17): 18944-18951.

[14] Howard A G, et al. MobileNets: Efficient convolutional neural networks for mobile vision applications[J]. arXiv preprint arXiv:1704.04861, 2017.

[15]Huawei Cloud Developer Community. Analysis of model quantization, pruning, and distillation techniques[EB/OL]. (2025-09-28). https://developer.huaweicloud.com/

This preprint was submitted under the following conditions:

- The authors declare that the necessary Terms of Free and Informed Consent of participants or patients in the research were obtained and are described in the manuscript, when applicable.
- The authors declare that the preparation of the manuscript followed the ethical norms of scientific communication.
- The authors declare that they are aware that they are solely responsible for the content of the preprint and that the deposit in SciELO Preprints does not mean any commitment on the part of SciELO, except its preservation and dissemination.
- The authors declare that the data, applications, and other content underlying the manuscript are referenced.
- The deposited manuscript is in PDF format.
- The authors declare that the research that originated the manuscript followed good ethical practices and that the necessary approvals from research ethics committees, when applicable, are described in the manuscript.
- The authors declare that once a manuscript is posted on the SciELO Preprints server, it can only be taken down on request to the SciELO Preprints server Editorial Secretariat, who will post a retraction notice in its place.
- The authors agree that the approved manuscript will be made available under a Creative Commons CC-BY license.
- The submitting author declares that the contributions of all authors and conflict of interest statement are included explicitly and in specific sections of the manuscript.
- The authors declare that the manuscript was not deposited and/or previously made available on another preprint server or published by a journal.
- If the manuscript is being reviewed or being prepared for publishing but not yet published by a journal, the authors declare that they have received authorization from the journal to make this deposit.
- The submitting author declares that all authors of the manuscript agree with the submission to SciELO Preprints.