

MCP Design Strategy for Plant-Factory Operation AI Agents

Byunghyun Ban

¹ Department of Music, Gyeongbuk National University, Andong, Republic of Korea

² Nanalab, Republic of Korea

* Correspondence: bhban@nanalab.kr

Abstract

Plant factories are increasingly equipped with sensors, cameras, automatic controllers, digital twins, and artificial intelligence models. However, integrating these components into a practical operation AI agent requires a safe interface between AI reasoning and facility-level control. This paper proposes an MCP design strategy for plant-factory operation AI agents. The focus is not on implementing a new autonomous agent or replacing existing control systems, but on designing the Model Context Protocol layer through which an agent can read plant-factory context, call diagnostic and simulation tools, follow standard operating prompts, and update supervisory operation parameters. In the proposed strategy, MCP resources represent sensor data, crop images, nutrient states, recipes, actuator states, logs, and digital-twin states; MCP tools represent diagnosis, prediction, simulation, and bounded parameter-update functions; and MCP prompts represent user-selectable standard operating procedure templates. The key argument is that MCP should not be used as a deterministic real-time controller for plant factories. Instead, it should serve as a supervisory parameter interface that modifies setpoints, recipes, schedules, thresholds, and policy parameters used by existing 24-hour automatic control systems. This paper presents resource URI schemes, tool risk classification, parameter-update tool schemas, prompt-resource-tool binding, safety tables, and closed-loop operation workflows for MCP-mediated plant-factory operation.

Keywords: Model Context Protocol, MCP, plant factory, AI agent, controlled-environment agriculture, supervisory control, parameter update, digital twin, nutrient solution, machine vision, safe closed-loop cultivation

1. Introduction

Plant factories have developed into highly instrumented controlled-environment agricultural systems. Modern plant factories may include RGB cameras, RGB-D cameras, hyperspectral imaging devices, pH and electrical conductivity sensors, ion-selective electrodes, temperature and humidity sensors, CO₂ sensors, LED lighting systems, HVAC devices, irrigation systems, nutrient tanks, dosing controllers, automatic control software, digital dashboards, and digital twins. Artificial intelligence has been applied to crop-state perception, fresh-weight estimation, stress detection, nutrient diagnosis, growth prediction, environmental control, digital twins, and product-level inspection [1-29].

However, the existence of individual AI models does not automatically produce a plant-factory operation agent. A model can estimate lettuce fresh weight [4-7,20,25-29], detect tip-burn or nutrient-deficiency symptoms [8-10,23,24], correct ion-selective electrode measurements [11-14], explore control policies [15], or support digital-twin simulation [16-18]. An operation agent, by contrast, must coordinate heterogeneous data and tools under operational constraints. It must read cultivation context, interpret plant and nutrient states, generate recommendations, simulate candidate interventions, request approval, update facility operation parameters, and produce auditable records.

In this paper, a plant-factory operation AI agent is defined as a supervisory AI system that can interpret plant-factory context, invoke diagnostic and simulation tools, generate operation recommendations, request bounded updates to supervisory control parameters through MCP tools, and produce auditable operation records under human and system-level safety constraints.

This paper does not assume that an AI agent directly performs real-time physical control of plant-factory devices. In commercial plant factories, lighting, HVAC, irrigation, CO₂ supply, and nutrient dosing are usually handled by automatic control systems that operate continuously. Therefore, the proposed MCP layer is positioned as a supervisory parameter interface, not as a deterministic real-time controller. The operation agent reads plant-factory context through MCP resources and modifies the direction of operation by updating setpoints, recipes, schedules, thresholds, and control-policy parameters through bounded MCP tools. The physical execution of these parameters remains the responsibility of existing control systems, PLCs, embedded controllers, SCADA systems, or facility-level automation software.

This position is especially appropriate for plant-factory operation because many agronomic decisions are evaluated over multi-hour, daily, crop-stage, or production-cycle timescales rather than sub-second control cycles. A lettuce canopy does not need a language model to switch a pump on for three seconds; it may need a supervisory system that notices slow growth, checks sensor and nutrient context, simulates recipe changes, and recommends a revised lighting or nutrient policy for the next several hours or days.

The purpose of this paper is to propose a design strategy for that MCP layer. This paper is a design-strategy and reference-architecture paper. It does not report a crop-growth experiment, a full MCP deployment, or a real facility implementation. Instead, it derives protocol-layer design requirements from plant-factory AI literature and proposes an MCP design strategy for connecting operation AI agents with plant-factory resources, diagnostic tools, standard operating prompts, simulation tools, parameter-update tools, and existing automatic control systems.

2. Background: AI Components Required by Plant-Factory Operation Agents

2.1 Machine vision and crop-state perception

Machine vision is one of the most mature AI components for plant-factory operation. A plant-factory agent must know the crop state before making operational recommendations. Previous studies have reviewed machine vision in plant factories and deep learning in controlled-environment agriculture, identifying growth monitoring, fresh-weight estimation, stress detection, robot operation, and product inspection as major applications [1,2].

Optical-flow analysis and machine learning have been used to predict lettuce growth from leaf movement, showing that plant growth can be interpreted as a temporal visual process [3]. RGB-D imaging, CNNs, segmentation, multimodal fusion, point-cloud completion, and production-oriented imaging setups have been used to estimate fresh weight, growth parameters, seedling status, and phenotypic traits [4-7,20-22,25-29]. These studies provide diagnostic and prediction functions that can be exposed as MCP tools.

2.2 Stress and disorder detection

Plant-factory agents should not only estimate biomass but also detect risk. Tip-burn detection studies show that deep-learning models can identify physiological disorders in indoor or plant-factory lettuce [8,9]. Nutrient-deficiency diagnosis has also been studied using multidimensional image analysis, hyperspectral imaging, and machine-vision methods [10,23,24]. These models can support early warning, anomaly triage, and intervention planning.

2.3 Nutrient-solution intelligence and sensor correction

Nutrient-solution management is a critical hidden layer of plant-factory operation. In hydroponic systems, pH and EC alone cannot fully describe individual ion balance. Ion-selective electrodes can measure individual ions, but they are affected by interference, drift, kinetic artifacts, and electrical artifacts. Machine-learning correction of ion interference, ODE-network modeling of nutrient-solution dynamics, deep-learning artifact removal from ISEs, and automated nutrient-solution management systems have been proposed to address these problems [11-14].

For an MCP-based operation agent, these studies imply that nutrient data should not be treated as raw sensor values only. The agent should read corrected nutrient resources, inspect sensor confidence, call nutrient-diagnosis tools, and simulate target-profile changes before requesting any update to the facility nutrient-management controller.

2.4 Environmental control and reinforcement learning

Plant-factory control is a multivariable problem. Lighting, HVAC, CO₂, airflow, irrigation, and nutrient conditions affect crop growth, energy use, physiological stress, and production schedules. Reinforcement learning has been applied to nonlinear and black-boxed greenhouse control, demonstrating a transferable method for controlled-

environment agriculture [15]. Although greenhouse systems and plant factories are not identical, both involve complex actuator interactions and incomplete system knowledge.

In the MCP design proposed here, reinforcement learning or optimization models are not treated as direct actuator drivers. Instead, they are treated as simulation or recommendation tools that may propose parameter updates for existing automatic controllers.

2.5 Digital twins and remote operation

Digital twins can represent plant-factory states in virtual environments. They can integrate sensor data, spatial layouts, crop models, environmental maps, and control logic. Studies on plant-factory digital twins and mixed-reality interfaces show that digital twins can support monitoring, simulation, and operator-centered remote control [16-18]. In the proposed architecture, the digital twin can act as both a resource server and a simulation tool server.

2.6 Dataset-driven product inspection

Plant-factory operation extends beyond cultivation. Commercial systems also require product grading, sorting, measurement, and quality assessment. The CongNaMul dataset demonstrates how crop-product inspection can be formulated as image classification, segmentation, decomposition, and physical feature measurement tasks [19]. Such methods can be exposed as postharvest inspection tools in an MCP-based operation workflow.

3. Position of This Paper

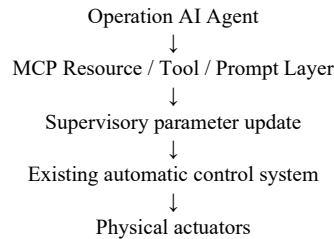
3.1 Design-strategy paper without implementation claim

This paper does not claim that a full plant-factory MCP system has been implemented or validated in a commercial facility. Its purpose is to define a protocol-layer design strategy. The contribution lies in mapping plant-factory AI components to MCP resources, tools, prompts, parameter-update workflows, and safety structures.

This framing matters because implementation papers and design-strategy papers should be evaluated differently. A plant-growth experiment asks whether a biological treatment improved yield. A software deployment paper asks whether a system worked in a facility. This paper asks how MCP should be designed if plant-factory operation AI agents are to interact safely with existing automatic control systems.

3.2 MCP as a supervisory parameter interface

The central claim of this paper is that MCP should expose plant-factory operation parameters, not replace facility controllers. The proposed MCP layer allows an operation AI agent to read cultivation context, interpret crop and nutrient states, simulate candidate changes, and update supervisory parameters such as setpoints, recipes, schedules, thresholds, and policy values. Continuous physical control remains within existing automatic control systems.



3.3 Difference between MCP operation and real-time control

MCP is an integration protocol that enables language-model applications to access context and external capabilities through structured client-server interactions [30,31]. It is not a deterministic real-time control protocol. Time-critical and safety-critical control must remain in PLCs, SCADA systems, embedded controllers, automatic climate controllers, nutrient controllers, and physical interlock systems.

Therefore, MCP tools should request bounded parameter updates. Lower-level controllers should enforce timing, hard limits, fail-safe behavior, and device-level execution.

4. MCP Feature Mapping for Plant-Factory Operation

MCP is suitable for plant-factory operation agents because its core primitives map naturally onto the needs of controlled-environment cultivation. Resources provide context, tools provide callable functions, and prompts provide workflow templates [30-34]. Resources are identified by URIs and may support subscription and list-change capabilities; tools include schema metadata and are model-invocable; prompts are exposed as structured templates intended for user-controlled selection or invocation.

MCP tools are designed to be model-controlled, but the official tools specification also emphasizes trust and safety, including human-in-the-loop confirmation and the ability to deny tool invocations [33]. This is especially important in plant factories because some tools may alter operational parameters that are later executed by physical controllers.

Table 1. MCP features and plant-factory design implications.

MCP feature	Role in MCP	Plant-factory operation implication
Base protocol	JSON-RPC-based client-server messaging	Operation agents communicate with plant-factory servers through structured requests and responses
Lifecycle	Initialization, capability negotiation, operation, shutdown	The agent discovers whether each server is read-only, diagnostic, simulation-capable, or parameter-update-capable
Resources	Contextual data identified by URIs	Sensor values, crop images, nutrient states, actuator states, recipes, logs, and digital-twin states
Resource subscription	Change notifications for selected resources	Environment, nutrient, alarm, and actuator-state updates can be monitored without direct device control
Tools	Callable functions exposed by servers	Diagnosis, prediction, simulation, recommendation, and bounded parameter-update functions
Tool schemas	JSON Schema-based argument validation	Zone, parameter, unit, range, duration, approval ID, and simulation ID can be enforced at the interface level
Prompts	Structured prompt templates	SOP-style workflows for reports, nutrient diagnosis, tip-burn triage, and control-action review
Authorization	Transport-level authorization for HTTP-based deployments	Role-based access can restrict who may read resources, run simulations, or approve parameter updates
Logging	Cross-cutting utility	Every resource read, tool call, approval, and parameter update can be recorded

5. Control Hierarchy for MCP-Based Plant-Factory Operation

The proposed MCP design separates AI reasoning from low-level physical execution.

Table 2. Control hierarchy for MCP-based plant-factory operation agents.

Layer	Function	MCP involvement	Example
Physical safety layer	Emergency stop, hard interlock, electrical/mechanical protection	None or indirect	CO2 upper-limit interlock, pump dry-run protection
Automatic control layer	Continuous 24-hour control	Not directly controlled by the agent	PID, PLC, SCADA, embedded climate/nutrient controllers
Supervisory parameter layer	Setpoint, recipe, schedule, threshold, policy update	Main MCP tool target	Change target PPFD, nutrient target profile, irrigation schedule
AI operation layer	Interpretation, recommendation, simulation, audit	MCP client behavior	Diagnose slow growth, recommend recipe update, request approval
Human governance layer	Approval, exception handling, accountability	Prompt/tool policy	Manager approves high-risk parameter changes

The core design principle is that MCP tools should update what the automatic controller should aim for, not micromanage how the controller drives physical actuators in real time.

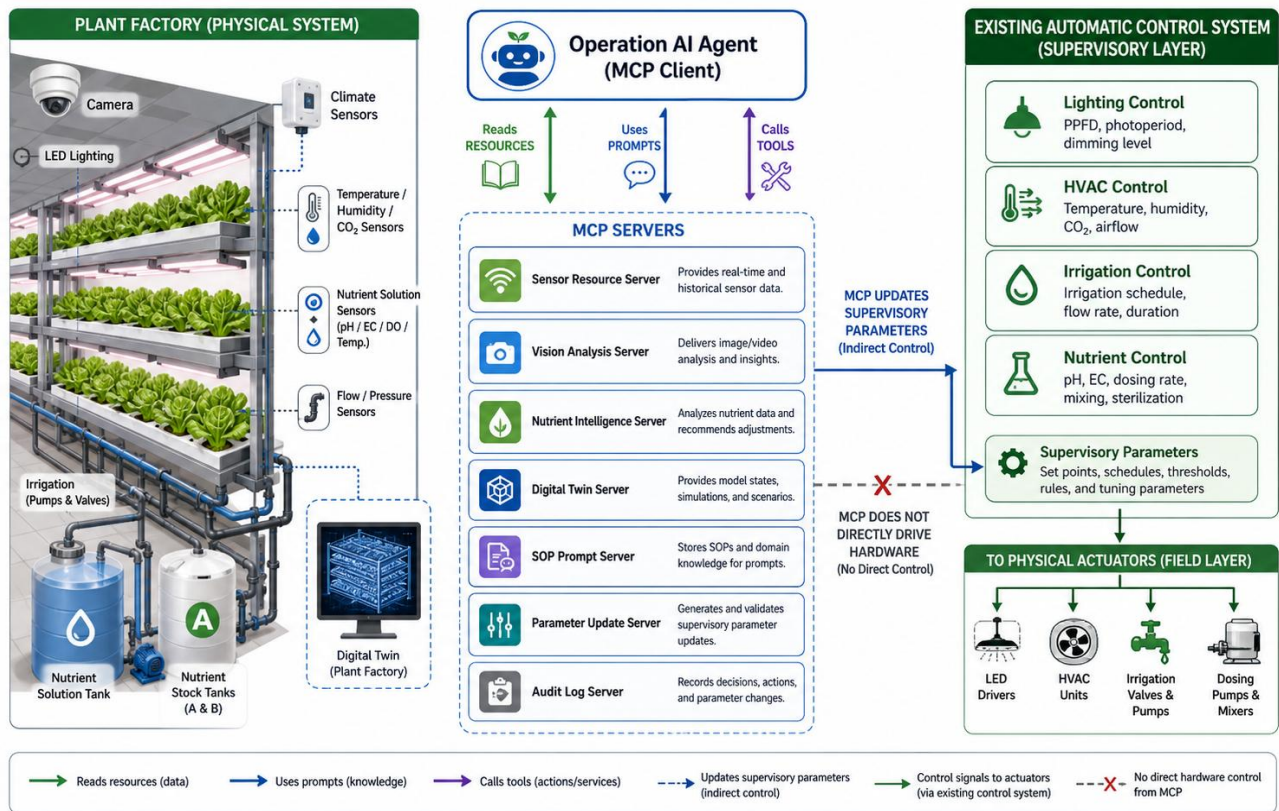


Figure 1. MCP architecture for plant-factory operation AI agents.

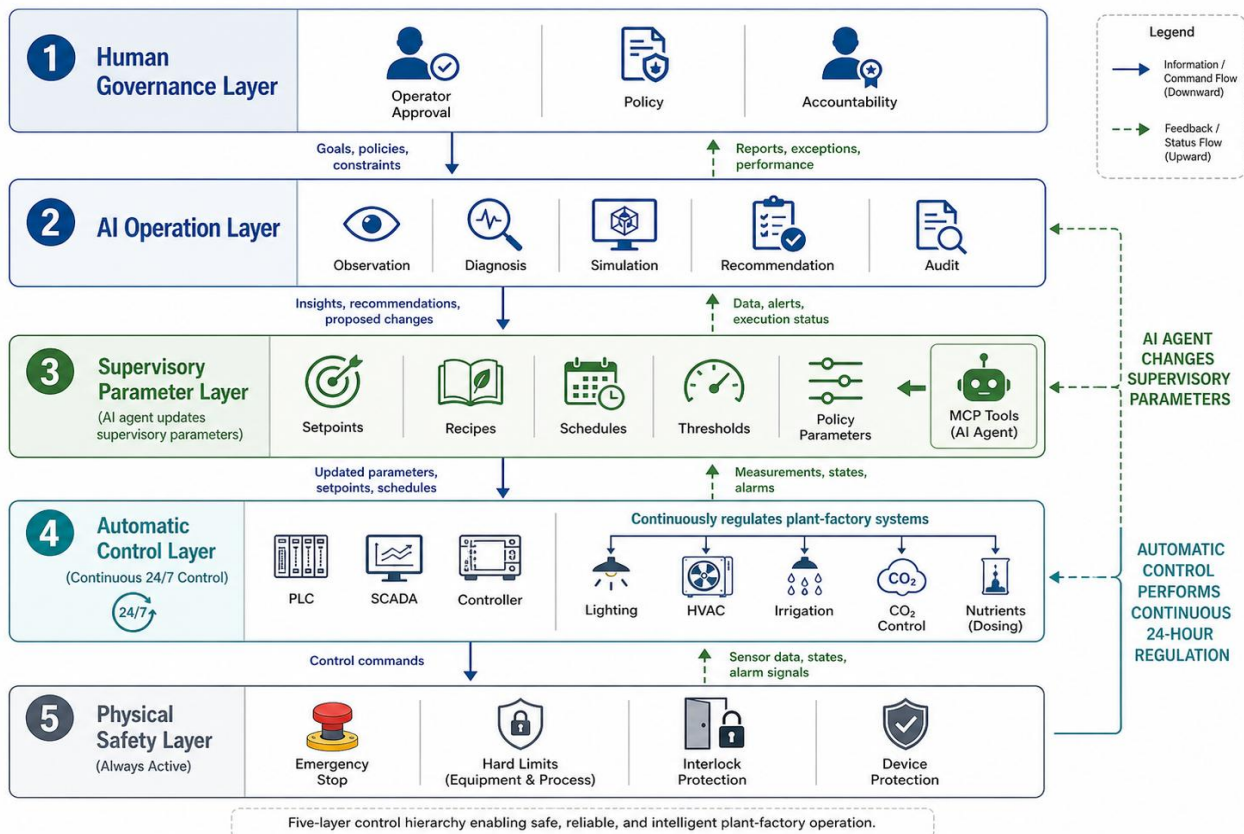


Figure 2. Five-layer control hierarchy for MCP-based plant-factory operation. The AI agent changes supervisory parameters, while the automatic control layer performs continuous 24-hour regulation and the physical safety layer remains always active.

6. From Plant-Factory AI Components to MCP Implications

Existing AI studies justify specific MCP resources and tools. The literature does not merely provide background; it determines which functions a plant-factory operation agent should be able to call.

Table 3. AI component to MCP implication mapping.

Existing AI component	Representative references	MCP implication
Machine-vision review and CEA deep learning	[1], [2]	General diagnostic and perception tool category
Leaf-motion growth prediction	[3]	predict_growth_from_time_series tool
RGB-D and CNN fresh-weight estimation	[4-7], [20], [25-29]	estimate_fresh_weight diagnostic tool
Tip-burn detection	[8], [9]	detect_tipburn anomaly tool
Nutrient-deficiency imaging	[10], [23], [24]	classify_nutrient_deficiency tool
ISE interference and artifact correction	[11], [13]	correct_ise_interference and remove_ise_artifacts tools
ODE nutrient-solution model	[12]	simulate_nutrient_profile_update tool
Nutrient-management system	[14]	Nutrient resources and target-profile update tools
Reinforcement-learning control	[15]	Policy simulation and parameter recommendation tools
Digital twin and MR interface	[16-18]	Digital-twin resources and simulation tools
Product inspection dataset	[19]	Postharvest classification and segmentation tools

7. MCP Resource Design Strategy

7.1 Resource principles

MCP resources allow servers to expose contextual data to clients. Each resource is identified by a URI [32]. In plant factories, resources should represent observable or historical states rather than executable commands.

Resources should be read-only from the agent perspective, hierarchically tied to site, zone, crop batch, device, and time, typed with MIME type or structured metadata when possible, versioned or timestamped for auditability, and protected by role-based access when sensitive.

7.2 Proposed resource URI scheme

A suggested convention is:

pf://{site_id}/zones/{zone_id}/{domain}/{object}/{time_or_state}

Examples:

pf://site-a/zones/z1/environment/current
pf://site-a/zones/z1/vision/rgb/latest
pf://site-a/zones/z1/vision/depth/latest
pf://site-a/zones/z1/nutrient-solution/current
pf://site-a/zones/z1/actuators/current
pf://site-a/batches/b17/crop-state/current
pf://site-a/batches/b17/recipe/current
pf://site-a/digital-twin/state
pf://site-a/logs/parameter-update/2026-05-04

7.3 Resource categories

Table 4. Proposed MCP resource categories for plant factories.

Resource category	Purpose	Example URI
Environment resources	Air and climate context	pf://site-a/zones/z1/environment/current
Vision resources	Crop images and image metadata	pf://site-a/zones/z1/vision/rgb/latest
Nutrient resources	Root-zone and solution context	pf://site-a/zones/z1/nutrient-solution/current

Crop-state resources	Growth stage and estimated traits	pf://site-a/batches/b17/crop-state/current
Recipe resources	Crop-stage target ranges	pf://site-a/batches/b17/recipe/current
Parameter resources	Current setpoints and schedules	pf://site-a/zones/z1/parameters/current
Digital-twin resources	Virtual facility and simulation state	pf://site-a/digital-twin/state
Log resources	Historical records and audit trails	pf://site-a/logs/parameter-update/2026-05-04

7.4 Resource validation rules

Resource servers should validate URI components:

- site_id must be registered.
- zone_id must exist.
- crop_batch_id must match active cultivation records.
- date values must follow ISO 8601 format.
- image resources must declare MIME type.
- nutrient resources must include timestamp and calibration metadata.
- sensitive resources must require authorization.
- stale resources must be marked with age or freshness metadata.

7.5 Resource subscription strategy

MCP resources can support subscription and list-change notifications [32]. In plant factories, subscriptions are useful for environment, nutrient, alarm, actuator-state, and parameter resources. However, high-frequency raw streams should be summarized by edge systems before being exposed to the AI agent. The agent does not need every sensor tick; it needs reliable context, trends, anomaly flags, and confidence indicators.

8. MCP Tool Design Strategy

8.1 Tool principles

MCP tools allow servers to expose callable functions to language models. Each tool is uniquely identified by a name and includes metadata describing its schema [33]. In plant factories, tools should be narrow, typed, bounded, auditable, and classified by operational risk.

The most important design change in this strategy is that high-risk MCP tools should be formulated as parameter-update tools, not immediate actuator commands.

Avoid:

```
turn_on_pump()
open_valve()
inject_co2_now()
activate_led()
```

Prefer:

```
update_lighting_setpoint_policy()
update_hvac_setpoint_policy()
update_irrigation_schedule_policy()
update_nutrient_target_profile()
submit_parameter_update_request()
simulate_parameter_update()
approve_parameter_update()
write_operation_audit_log()
```

8.2 Risk classification

Table 5. Risk classification of MCP tools for plant-factory operation agents.

Risk level	Tool category	Example tools	Physical implication	Approval
Level 0	Read-only summarization	summarize_environment, summarize_growth_log	No physical effect	Not required

Level 1	Diagnosis	detect_tipburn, diagnose_nutrient_status	No physical effect	Not required
Level 2	Simulation	simulate_lighting_policy, simulate_nutrient_profile_update	No physical effect	Not required
Level 3	Recommendation	generate_parameter_update_plan	No physical effect until approved	Review recommended
Level 4	Bounded parameter update	update_lighting_setpoint_policy, update_irrigation_schedule_policy	Existing controller changes operation gradually	Approval or policy-bound auto-approval
Level 5	High-risk parameter update	update_co2_target_policy, update_nutrient_target_profile, update_hvac_setpoint_policy	Facility controller may alter chemical or climate conditions	Mandatory approval

Even Level 4 and Level 5 tools are not direct actuator commands. They update parameters that existing controllers later execute.

8.3 Diagnostic tools

Examples:

```
estimate_fresh_weight(zone_id, crop_batch_id, image_uri)
detect_tipburn(zone_id, image_uri)
classify_nutrient_deficiency(image_uri, crop_stage)
diagnose_nutrient_status(nutrient_resource_uri)
summarize_environment(environment_resource_uri, time_window)
```

These tools can be based on machine-vision, hyperspectral, RGB-D, point-cloud, and multimodal studies [1-10,20-29].

8.4 Nutrient-intelligence tools

Examples:

```
correct_ise_interference(raw_ise_resource_uri)
remove_ise_artifacts(raw_ise_resource_uri)
estimate_ion_balance(nutrient_resource_uri)
simulate_nutrient_profile_update(current_solution_uri, target_recipe_uri)
recommend_nutrient_target_profile(batch_id, nutrient_state_uri)
```

These tools are informed by ISE correction, ODE nutrient modeling, artifact removal, and nutrient-solution management research [11-14].

8.5 Digital-twin and simulation tools

Examples:

```
update_digital_twin(sensor_snapshot_uri)
simulate_environment_policy_update(zone_id, control_candidate)
simulate_growth_response(batch_id, parameter_update_plan)
compare_parameter_update_scenarios(batch_id, objective)
```

These tools support simulation before parameter update [16-18].

8.6 Parameter-update tool schema examples

MCP uses JSON Schema for validation throughout the protocol, and tools include inputSchema metadata [31,33]. Therefore, plant-factory MCP tools should force numerical, bounded, auditable arguments rather than accepting vague commands such as "increase light a little."

Example 1. Lighting setpoint policy update

```
{
  "name": "update_lighting_setpoint_policy",
  "description": "Update the lighting setpoint policy used by the existing automatic lighting controller.",
  "inputSchema": {
    "type": "object",
    "properties": {
      "site_id": { "type": "string" },
      "zone_id": { "type": "string" },
      "crop_batch_id": { "type": "string" },

```

```

    "target_ppfd": { "type": "number", "minimum": 50, "maximum": 350 },
    "unit": { "type": "string", "enum": ["umol_m2_s"] },
    "photoperiod_hours": { "type": "number", "minimum": 6, "maximum": 24 },
    "ramp_minutes": { "type": "integer", "minimum": 0, "maximum": 180 },
    "effective_from": { "type": "string", "format": "date-time" },
    "duration_hours": { "type": "number", "minimum": 1, "maximum": 168 },
    "simulation_id": { "type": "string" },
    "approval_id": { "type": "string" },
    "reason": { "type": "string" }
  },
  "required": [
    "site_id", "zone_id", "crop_batch_id", "target_ppfd", "unit",
    "photoperiod_hours", "effective_from", "duration_hours",
    "simulation_id", "approval_id", "reason"
  ],
  "additionalProperties": false
}

```

This tool does not switch LEDs directly. It updates the target PPFD, photoperiod, ramp time, and effective period used by the existing lighting controller.

Example 2. Nutrient target profile update

```

{
  "name": "update_nutrient_target_profile",
  "description": "Update the nutrient target profile used by the existing nutrient-solution management controller.",
  "inputSchema": {
    "type": "object",
    "properties": {
      "site_id": { "type": "string" },
      "zone_id": { "type": "string" },
      "tank_id": { "type": "string" },
      "crop_batch_id": { "type": "string" },
      "target_ph": { "type": "number", "minimum": 4.5, "maximum": 7.5 },
      "target_ec": { "type": "number", "minimum": 0.2, "maximum": 4.0 },
      "ion_targets_mg_l": {
        "type": "object",
        "properties": {
          "NO3": { "type": "number", "minimum": 0, "maximum": 300 },
          "K": { "type": "number", "minimum": 0, "maximum": 300 },
          "Ca": { "type": "number", "minimum": 0, "maximum": 250 },
          "Mg": { "type": "number", "minimum": 0, "maximum": 100 }
        }
      },
      "additionalProperties": false
    },
    "effective_from": { "type": "string", "format": "date-time" },
    "simulation_id": { "type": "string" },
    "approval_id": { "type": "string" },
    "reason": { "type": "string" }
  },
  "required": [
    "site_id", "zone_id", "tank_id", "crop_batch_id", "target_ph", "target_ec",
    "ion_targets_mg_l", "effective_from", "simulation_id", "approval_id", "reason"
  ],
  "additionalProperties": false
}

```

This tool does not directly command dosing pumps. It updates the target nutrient profile that the existing nutrient-management controller will follow.

Example 3. Irrigation schedule policy update

```

{
  "name": "update_irrigation_schedule_policy",
  "description": "Update the irrigation schedule policy used by the existing irrigation controller.",
  "inputSchema": {
    "type": "object",
    "properties": {
      "site_id": { "type": "string" },
      "zone_id": { "type": "string" },

```

```

"crop_batch_id": { "type": "string" },
"irrigation_interval_minutes": { "type": "integer", "minimum": 10, "maximum": 720 },
"irrigation_duration_seconds": { "type": "integer", "minimum": 5, "maximum": 600 },
"active_window_start": { "type": "string", "pattern": "^[01][0-9]2[0-3]:[0-5][0-9]$" },
"active_window_end": { "type": "string", "pattern": "^[01][0-9]2[0-3]:[0-5][0-9]$" },
"effective_from": { "type": "string", "format": "date-time" },
"simulation_id": { "type": "string" },
"approval_id": { "type": "string" },
"reason": { "type": "string" }
},
"required": [
"site_id", "zone_id", "crop_batch_id", "irrigation_interval_minutes",
"irrigation_duration_seconds", "active_window_start", "active_window_end",
"effective_from", "simulation_id", "approval_id", "reason"
],
"additionalProperties": false
}
}

```

This tool updates the irrigation schedule policy. Pump operation remains the responsibility of the irrigation controller.

8.7 Error handling and rejection logic

MCP tools support protocol-level errors and tool-execution errors [33]. Plant-factory parameter-update tools should reject:

- values outside schema limits
- missing approval_id for high-risk updates
- missing simulation_id when simulation is required
- stale sensor resources
- inactive crop_batch_id
- zone_id not controlled by the server
- parameter changes conflicting with facility interlocks
- requests made by unauthorized roles

Error messages should be actionable so that the agent can revise the proposal rather than bypass the restriction.

Parameter-oriented closed-loop workflow for plant-factory operation AI agents

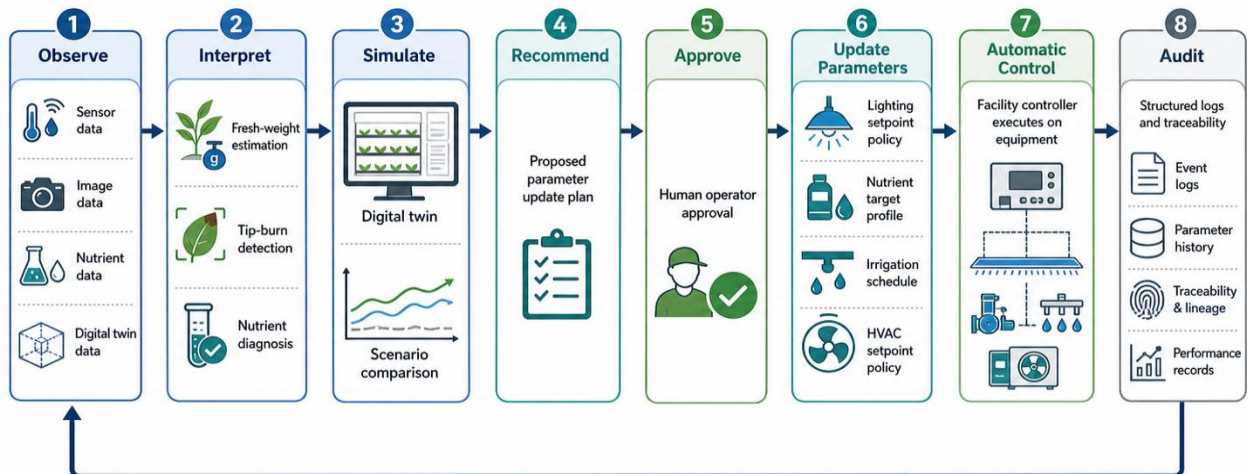


Figure 3. *Parameter-oriented closed-loop workflow for plant-factory operation AI agents. The agent observes context, interprets state, simulates candidate changes, recommends a parameter update, requests approval, updates supervisory parameters, and audits outcomes while existing controllers perform continuous physical control.*

9. MCP Prompt Design Strategy

9.1 SOP prompts as workflow templates

MCP prompts provide structured messages and instructions for interacting with language models, and they are designed to be user-controlled [34]. In this paper, MCP prompts are not treated as autonomous execution commands. They are treated as user-selectable or policy-selectable workflow templates that guide the operation agent in choosing relevant resources and permitted tools.

For example, a nutrient-imbalance diagnosis prompt may allow the agent to read nutrient resources, call ISE-correction tools, and run nutrient-profile simulations. It should not permit direct modification of nutrient target profiles unless a separate approval workflow is completed.

9.2 Proposed prompt library

Table 6. SOP-style MCP prompts for plant-factory operation.

Prompt name	Purpose	Required resources	Allowed tools
daily_operation_report	Summarize facility state	Environment, crop, nutrient, parameter, log resources	Summarization tools
tipburn_risk_triage	Diagnose tip-burn risk	RGB image, humidity, VPD, crop stage	Vision and environment tools
nutrient_imbalance_diagnosis	Interpret nutrient imbalance	pH, EC, ISE, recipe, history	ISE correction, nutrient simulation
harvest_readiness_check	Assess harvest timing	Crop images, growth estimates, schedule	Fresh-weight estimation
control_action_review	Review proposed parameter update	Current state, simulation result, recipe	Simulation and risk tools
postharvest_quality_report	Grade harvested products	Product images, batch info	Classification and segmentation tools

9.3 Prompt-resource-tool binding

Each prompt should specify allowed resources and tools. For example:

Prompt: nutrient_imbalance_diagnosis

Goal:

Diagnose whether the nutrient solution deviates from the crop-stage recipe.

Required resources:

- current nutrient-solution resource
- crop-stage recipe resource
- last 7 days of nutrient logs

Allowed tools:

- correct_ise_interference
- remove_ise_artifacts
- estimate_ion_balance
- simulate_nutrient_profile_update

Forbidden tools:

- update_nutrient_target_profile
- update_lighting_setpoint_policy
- update_hvac_setpoint_policy

Output:

- diagnosis
- confidence
- likely causes
- recommended simulation
- whether human review is required

Prompt-resource-tool binding reduces workflow drift and prevents an agent from using unrelated high-risk tools during a diagnostic task.

10. Parameter-Oriented Closed-Loop Workflow

A plant-factory operation agent should follow a structured workflow that separates observation, interpretation, simulation, recommendation, approval, parameter update, automatic control, and audit.

10.1 Observe

The agent reads MCP resources:

```
environment/current
vision/rgb/latest
vision/depth/latest
nutrient-solution/current
crop-state/current
parameters/current
actuators/current
digital-twin/state
```

This step is read-only.

10.2 Interpret

The agent calls diagnostic tools:

```
estimate_fresh_weight()
detect_tipburn()
diagnose_nutrient_status()
summarize_environment()
```

Outputs should include confidence, uncertainty, and resource timestamps.

10.3 Simulate

Before parameter updates, the agent calls simulation tools:

```
simulate_lighting_policy()
simulate_hvac_setpoint_policy()
simulate_nutrient_profile_update()
simulate_irrigation_schedule_policy()
simulate_growth_response()
```

The simulation result should produce a `simulation_id`.

10.4 Recommend

The agent generates a bounded parameter-update plan:

```
Recommended parameter update:
- Zone: Z1
- Parameter: target_ppfd
- Current value: 180 umol m-2 s-1
- Proposed value: 210 umol m-2 s-1
- Duration: 48 hours
- Ramp: 60 minutes
- Reason: predicted biomass is below schedule, no current tip-burn signal, nutrient status stable
- Required approval: manager
```

10.5 Approve

A human operator or predefined governance policy approves the proposed parameter update. The approval creates an `approval_id`.

10.6 Update parameters

The agent calls a parameter-update MCP tool such as:

```
update_lighting_setpoint_policy()
```

```

update_nutrient_target_profile()
update_irrigation_schedule_policy()
update_hvac_setpoint_policy()

```

10.7 Automatic control

The facility automatic control system continuously executes the updated parameters through its own control loops. MCP does not replace this layer.

10.8 Audit

All resources, diagnostics, simulations, recommendations, approvals, parameter updates, and post-update states are logged.

```

resources_read
tools_called
prompt_used
simulation_id
recommendation
approval_id
parameter_update_request
controller_acknowledgement
post_update_state

```

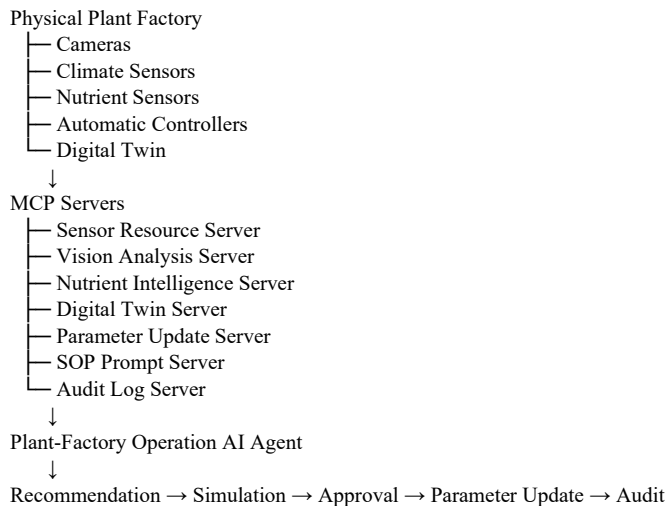
11. Reference MCP Server Architecture

A plant-factory MCP system should be modular rather than monolithic.

Table 7. Proposed MCP server modules.

MCP server	Main function	MCP primitives
Sensor Resource Server	Expose climate, nutrient, and device states	Resources
Vision Analysis Server	Estimate traits and detect visual disorders	Tools
Nutrient Intelligence Server	Correct ISE data and simulate nutrient profiles	Resources, tools
Digital Twin Server	Provide virtual state and simulations	Resources, tools
Parameter Update Server	Update setpoints, recipes, schedules, and thresholds	Tools
SOP Prompt Server	Provide operation workflow templates	Prompts
Audit Log Server	Record all actions and outcomes	Resources, tools

The architecture can be summarized as:



During initialization, MCP clients and servers establish protocol compatibility, exchange capabilities, and share implementation details. Capability negotiation can declare whether a server provides resources, prompts, tools, subscriptions, list-change notifications, or logging [35]. In plant factories, capability metadata should also disclose whether a server is read-only, diagnostic, simulation-capable, or parameter-update-capable.

Example domain-specific capability metadata:

```
{
  "server_role": "parameter_update",
  "supported_zones": ["Z1", "Z2"],
  "parameter_domains": ["lighting", "irrigation", "nutrient"],
  "risk_levels": [3, 4, 5],
  "requires_human_approval": true,
  "supports_simulation_reference": true,
  "writes_audit_log": true
}
```

12. Case Scenarios

12.1 Lettuce growth monitoring and lighting policy update

Table 8. MCP workflow for growth monitoring and lighting policy update.

Step	MCP primitive	Example
Observe	Resource	pf://site-a/zones/z1/vision/rgb/latest
Estimate	Tool	estimate_fresh_weight(image_uri)
Context check	Resource	environment/current, nutrient-solution/current, recipe/current
Simulate	Tool	simulate_lighting_policy(target_ppfd=210)
Recommend	Prompt	control_action_review
Approve	Governance	Human approval creates approval_id
Update	Tool	update_lighting_setpoint_policy()
Execute	Existing controller	Lighting controller follows new setpoint policy
Audit	Resource/tool	write_operation_audit_log()

The agent does not directly control LED hardware. It updates the lighting policy that the facility controller follows.

12.2 Tip-burn risk triage and climate setpoint review

Table 9. MCP workflow for tip-burn risk triage.

Step	MCP primitive	Example
Observe	Resource	RGB image, humidity, VPD, airflow, crop stage
Diagnose	Tool	detect_tipburn(image_uri)
Context check	Resource	environment/current, growth-rate/history
Simulate	Tool	simulate_hvac_setpoint_policy()
Recommend	Prompt	tipburn_risk_triage
Approve	Governance	Required for HVAC policy update
Update	Tool	update_hvac_setpoint_policy()
Execute	Existing controller	HVAC controller follows updated setpoint policy
Audit	Resource/tool	Record diagnosis, simulation, approval, update

12.3 Nutrient imbalance diagnosis and target-profile update

Table 10. MCP workflow for nutrient target-profile update.

Step	MCP primitive	Example
Observe	Resource	pH, EC, ISE, tank status, nutrient history
Correct	Tool	correct_ise_interference()
Diagnose	Tool	estimate_ion_balance()
Simulate	Tool	simulate_nutrient_profile_update()
Recommend	Prompt	nutrient_imbalance_diagnosis
Approve	Governance	Mandatory approval for target-profile update
Update	Tool	update_nutrient_target_profile()
Execute	Existing controller	Nutrient controller computes and performs dosing
Audit	Resource/tool	Record target profile, simulation, approval, controller acknowledgement

This scenario shows why parameter-oriented MCP is safer than direct dosing. The MCP tool updates the target profile; the existing nutrient controller handles dosing, mixing, pH/EC cutoffs, and device-level safety.

12.4 Digital-twin-based remote operation

The agent updates a digital twin using sensor resources and image data [16-18]. It compares parameter-update scenarios and presents predicted effects to the operator. The operator approves one plan through a dashboard or mixed-reality interface. The MCP layer then sends bounded parameter updates to the appropriate server.

12.5 Postharvest product inspection

After harvest, the agent reads product-image resources and calls inspection tools. Dataset-driven methods such as classification, segmentation, and physical measurement can support grading and quality reports [19]. This expands MCP-mediated operation beyond cultivation into postharvest workflow management.

13. Safety, Security, and Governance

13.1 Safety risks and mitigations

Table 11. Safety risks and mitigations in parameter-oriented MCP control.

Risk	Example	MCP-level mitigation	Facility-level mitigation
Excessive lighting target	Agent proposes too high PPFD	Schema max, crop-stage recipe check, simulation ID	Lighting controller upper bound
Nutrient target error	Wrong ion target profile	ISE correction, ODE simulation, approval token	Dosing controller limit, mixing delay, EC/pH cutoff
Irrigation over-adjustment	Too frequent irrigation schedule	Minimum interval schema, policy validation	Pump runtime limit, drain/overflow sensor
HVAC policy conflict	Heating and cooling conflict	Scenario simulation, policy consistency check	PLC/SCADA interlock
CO2 target risk	Excessive CO2 target	High-risk approval, target upper bound	CO2 controller hard limit, ventilation interlock
Sensor drift	Wrong resource values drive wrong update	Sensor confidence resource, calibration log	Manual calibration, redundant sensors
Prompt injection	Malicious instruction in logs or notes	Prompt-tool binding, tool allowlist	Operator confirmation
Unauthorized parameter update	Wrong user changes recipe	Authorization, role-based approval	Facility account control, physical lockout

13.2 Authorization and role-based access

MCP provides authorization capabilities at the transport level for HTTP-based transports, and authorization is optional for MCP implementations [36]. This means MCP authorization can support restricted access, but it is not sufficient as a plant-factory safety mechanism by itself. Role-based access, approval workflows, actuator-level limits, and emergency interlocks must be implemented outside or below the MCP layer.

Table 12. Suggested access roles for plant-factory MCP deployments.

Role	Resource access	Tool access
Researcher	Read crop and environment data	Analysis and simulation tools only
Operator	Read facility state	Recommendation and low-risk parameter tools
Manager	Read all operational data	Approve high-risk parameter updates
Technician	Read device state	Maintenance tools
Safety system	Independent access	Override and emergency stop outside MCP

13.3 Separation of recommendation and parameter update

The agent should produce recommendations before updating parameters. Execution should be a separate, approved step. This separation is necessary because language models may be uncertain, may misinterpret context, or may be affected by malicious resource content.

13.4 Prompt injection and tool misuse

Plant-factory MCP systems may expose images, logs, documents, and operator notes as resources. These resources may contain misleading or adversarial text. Therefore, tools should not trust arbitrary resource content. Servers must validate inputs independently of the agent reasoning. Prompt templates should define allowed resources and allowed tools.

13.5 Audit and accountability

Every high-risk parameter update should be explainable after the fact. The audit log should answer:

- What did the agent observe?
- Which resources were read?
- Which tools were called?
- What simulation was performed?
- What parameter update was recommended?
- Who approved it?
- What parameter was changed?
- Did the controller acknowledge it?
- What changed afterward?

Auditability is required for debugging, operator trust, insurance, regulatory compliance, and incident analysis.

14. Discussion

14.1 Why parameter-oriented MCP is appropriate for plant factories

Plant-factory operation is not mostly about instantaneous control decisions. Many operational decisions concern target ranges, recipes, schedules, and trajectories over hours or days. This makes MCP suitable as a supervisory operation layer. The agent can interpret rich context, reason over crop-stage goals, simulate candidate changes, and update parameters that existing controllers execute continuously.

14.2 Why MCP should not replace automatic control systems

MCP is useful because it standardizes how an AI agent accesses resources, prompts, and tools. It should not be treated as a replacement for PLCs, SCADA systems, embedded controllers, safety relays, or facility automation software. The lower-level control system must retain responsibility for deterministic timing, hardware protection, fail-safe behavior, and emergency response.

14.3 Limitations of this design-strategy paper

This paper does not include a working MCP implementation, real plant-factory deployment, or crop-growth experiment. Therefore, it does not evaluate latency, server reliability, operator acceptance, or biological outcomes. The proposed architecture should be understood as a reference design. Future work should implement prototype MCP servers and evaluate them in simulated and real plant-factory environments.

14.4 From AI monitoring to MCP-mediated biological manufacturing

Plant factories are not merely automated farms. They are biological manufacturing systems in which living crops respond to environmental, chemical, and temporal conditions. The role of an MCP-mediated operation agent is to connect perception, nutrient intelligence, digital twins, parameter-update tools, SOP prompts, and human governance. The goal is not unconstrained autonomy, but safe and auditable closed-loop cultivation support.

15. Conclusion

This paper proposed an MCP design strategy for plant-factory operation AI agents. The central argument is that MCP should serve as a supervisory parameter interface rather than a real-time physical controller. In the proposed strategy, MCP resources represent read-only plant-factory context, MCP tools represent diagnostic, simulation, and bounded parameter-update functions, and MCP prompts represent user-selectable standard operating procedure templates.

Existing plant-factory AI studies provide the functional modules required by such agents: machine vision for crop-state perception, deep learning for stress detection, nutrient-solution intelligence for root-zone diagnosis, reinforcement learning for control-policy exploration, digital twins for simulation and remote operation, and product datasets for postharvest inspection [1-29]. MCP can connect these modules into an operation layer by allowing the agent to observe, interpret, simulate, recommend, obtain approval, update parameters, and audit outcomes.

The proposed design deliberately separates AI reasoning from continuous physical control. The agent updates setpoints, recipes, schedules, thresholds, and policy parameters. Existing automatic control systems execute those parameters through their own control loops and safety mechanisms. This division is appropriate for plant-factory operation because many agronomic decisions occur over multi-hour, daily, or crop-stage timescales rather than sub-second cycles.

Future work should implement prototype MCP servers for plant factories, including sensor resource servers, vision-analysis tool servers, nutrient-intelligence servers, digital-twin servers, parameter-update servers, SOP prompt servers, and audit-log servers. Evaluation should include not only model accuracy but also safety, auditability, operator trust, cross-facility generalization, failure recovery, and biological outcomes.

References

- [1] Tian, Z.; Ma, W.; Yang, Q.; Duan, F. Application Status and Challenges of Machine Vision in Plant Factory-A Review. *Information Processing in Agriculture*, 2022, 9, 195-211. doi: 10.1016/j.inpa.2021.06.003.
- [2] Ojo, M. O.; Zahid, A. Deep Learning in Controlled Environment Agriculture: A Review of Recent Advancements, Challenges and Prospects. *Sensors*, 2022, 22, 7965. doi: 10.3390/s22207965.
- [3] Nagano, S.; Moriyuki, S.; Wakamori, K.; Mineno, H.; Fukuda, H. Leaf-Movement-Based Growth Prediction Model Using Optical Flow Analysis and Machine Learning in Plant Factory. *Frontiers in Plant Science*, 2019, 10, 227. doi: 10.3389/fpls.2019.00227.
- [4] Lin, Z.; Fu, R.; Ren, G.; Zhong, R.; Ying, Y.; Lin, T. Automatic Monitoring of Lettuce Fresh Weight by Multi-Modal Fusion Based Deep Learning. *Frontiers in Plant Science*, 2022, 13, 980581. doi: 10.3389/fpls.2022.980581.
- [5] Moon, T.; Choi, W.-J.; Jang, S.-H.; Choi, D.-S.; Oh, M.-M. Growth Analysis of Plant Factory-Grown Lettuce by Deep Neural Networks Based on Automated Feature Extraction. *Horticulturae*, 2022, 8, 1124. doi: 10.3390/horticulturae8121124.
- [6] Tan, J.; Hou, J.; Xu, W.; Zheng, H.; Gu, S.; Zhou, Y.; Qi, L.; Ma, R. PosNet: Estimating Lettuce Fresh Weight in Plant Factory Based on Oblique Image. *Computers and Electronics in Agriculture*, 2023, 213, 108263. doi: 10.1016/j.compag.2023.108263.
- [7] Kim, J. S. G.; Moon, S.; Park, J.; Kim, T.; Chung, S. Development of a Machine Vision-Based Weight Prediction System of Butterhead Lettuce Using Deep Learning Models for Industrial Plant Factory. *Frontiers in Plant Science*, 2024, 15, 1365266. doi: 10.3389/fpls.2024.1365266.
- [8] Hamidon, M. H.; Ahamed, T. Detection of Tip-Burn Stress on Lettuce Grown in an Indoor Environment Using Deep Learning Algorithms. *Sensors*, 2022, 22, 7251. doi: 10.3390/s22197251.
- [9] Kumaratenna, K. P. S.; Cho, Y. Y. Detection of Tipburn Stress on Lettuce Grown in a Plant Factory Using Artificial Intelligence Models. *Horticultural Science and Technology*, 2024, 42, 711-724.

- [10] Xie, J.; et al. Exploring Nutrient Deficiencies in Lettuce Crops: Utilizing Advanced Multidimensional Image Analysis for Precision Diagnosis. *Sensors*, 2025, 25, 1957. doi: 10.3390/s25071957.
- [11] Ban, B.; Ryu, D.; Lee, M. Machine Learning Approach to Remove Ion Interference Effect in Agricultural Nutrient Solutions. *International Conference on Information and Communication Technology Convergence*, 2019, 1156-1161. doi: 10.1109/ICTC46691.2019.8939812.
- [12] Ban, B.; Lee, M.; Ryu, D. ODE Network Model for Nonlinear and Complex Agricultural Nutrient Solution System. *International Conference on Information and Communication Technology Convergence*, 2019, 996-1001. doi: 10.1109/ICTC46691.2019.8939946.
- [13] Ban, B.; Ryu, D.; Lee, M. Deep Learning Method to Remove Chemical, Kinetic and Electric Artifacts on ISEs. *International Conference on Information and Communication Technology Convergence*, 2020, 1242-1246; Preprints 2020, 2020050381. doi: 10.20944/preprints202005.0381.v2.
- [14] Ban, B.; Lee, J.; Ryu, D.; Lee, M.; Eom, T. D. Nutrient Solution Management System for Smart Farms and Plant Factory. *International Conference on Information and Communication Technology Convergence*, 2020, 1537-1542. doi: 10.1109/ICTC49870.2020.9289192.
- [15] Ban, B.; Kim, S. Control of Nonlinear, Complex and Black-Boxed Greenhouse System with Reinforcement Learning. *International Conference on Information and Communication Technology Convergence*, 2017, 913-918. doi: 10.1109/ICTC.2017.8190813.
- [16] Zhang, Z.; Zhu, Z.; Gao, G.; Qu, D.; Zhong, J.; Jia, D.; Du, X.; Yang, X.; Pan, S. Design and Research of Digital Twin System for Multi-Environmental Variable Mapping in Plant Factory. *Computers and Electronics in Agriculture*, 2023, 213, 108243. doi: 10.1016/j.compag.2023.108243.
- [17] Chen, K.; Zhao, B.; Zhou, H.; Zhou, L.; Niu, K.; Jin, X.; Li, R.; Yuan, Y.; Zheng, Y. Digital Twins in Plant Factory: A Five-Dimensional Modeling Method for Plant Factory Transplanter Digital Twins. *Agriculture*, 2023, 13, 1336. doi: 10.3390/agriculture13071336.
- [18] Ban, B. Mixed Reality Interface for Digital Twin of Plant Factory. *arXiv*, 2022, arXiv:2211.00597. doi: 10.48550/arXiv.2211.00597.
- [19] Ban, B.; Ryu, D.; Hwang, S.-w. CongNaMul: A Dataset for Advanced Image Processing of Soybean Sprouts. *International Conference on Information and Communication Technology Convergence*, 2023, 1892-1897; arXiv:2308.15690.
- [20] Zhang, L.; Xu, Z.; Xu, D.; Ma, J.; Chen, Y.; Fu, Z. Growth Monitoring of Greenhouse Lettuce Based on a Convolutional Neural Network. *Horticulture Research*, 2020, 7, 124. doi: 10.1038/s41438-020-00345-6.
- [21] Chang, C.-L.; Chung, S.-C.; Fu, W.-L.; Huang, C.-C. Artificial Intelligence Approaches to Predict Growth, Harvest Day, and Quality of Lettuce (*Lactuca sativa* L.) in an IoT-Enabled Greenhouse System. *Biosystems Engineering*, 2021, 212, 77-105. doi: 10.1016/j.biosystemseng.2021.09.015.
- [22] Lou, M.; Lu, J.; Wang, L.; Jiang, H.; Zhou, M. Growth Parameter Acquisition and Geometric Point Cloud Completion of Lettuce. *Frontiers in Plant Science*, 2022, 13, 947690. doi: 10.3389/fpls.2022.947690.
- [23] Pandey, P.; Veazie, P.; Whipker, B. E.; Young, S. N. Predicting Foliar Nutrient Concentrations and Nutrient Deficiencies of Hydroponic Lettuce Using Hyperspectral Imaging. *Biosystems Engineering*, 2023, 230, 458-469. doi: 10.1016/j.biosystemseng.2023.05.005.
- [24] Lu, J.; Peng, K.; Wang, Q.; Sun, C. Lettuce Plant Trace-Element-Deficiency Symptom Identification via Machine Vision Methods. *Agriculture*, 2023, 13, 1614. doi: 10.3390/agriculture13081614.
- [25] Xu, D.; Chen, J.; Li, B.; Ma, J. Improving Lettuce Fresh Weight Estimation Accuracy through RGB-D Fusion. *Agronomy*, 2023, 13, 2617. doi: 10.3390/agronomy13102617.
- [26] Ojo, M. O.; Zahid, A.; Masabni, J. G. Estimating Hydroponic Lettuce Phenotypic Parameters for Efficient Resource Allocation. *Computers and Electronics in Agriculture*, 2024, 218, 108642. doi: 10.1016/j.compag.2024.108642.
- [27] Islam, S.; Reza, M. N.; Chowdhury, M.; Ahmed, S.; Lee, K.-H.; Ali, M.; Cho, Y. J.; Noh, D. H.; Chung, S.-O. Detection and Segmentation of Lettuce Seedlings from Seedling-Growing Tray Imagery Using an Improved Mask R-CNN Method. *Smart Agricultural Technology*, 2024, 8, 100455. doi: 10.1016/j.atech.2024.100455.
- [28] Hosoda, Y.; Tada, T.; Goto, H. Lettuce Fresh Weight Prediction in a Plant Factory Using Plant Growth Models. *IEEE Access*, 2024, 12, 97226-97234. doi: 10.1109/ACCESS.2024.3423455.
- [29] Hou, L.; Zhu, Y.; Wang, M.; Wei, N.; Dong, J.; Tao, Y.; Zhou, J.; Zhang, J. Multimodal Data Fusion for Precise Lettuce Phenotype Estimation Using Deep Learning Algorithms. *Plants*, 2024, 13, 3217. doi: 10.3390/plants13223217.
- [30] Model Context Protocol. Specification. Version 2025-11-25, 2025. URL: <https://modelcontextprotocol.io/specification/2025-11-25>.
- [31] Model Context Protocol. Base Protocol: Overview. Version 2025-11-25, 2025. URL: <https://modelcontextprotocol.io/specification/2025-11-25/basic>.
- [32] Model Context Protocol. Server Features: Resources. Version 2025-11-25, 2025. URL: <https://modelcontextprotocol.io/specification/2025-11-25/server/resources>.
- [33] Model Context Protocol. Server Features: Tools. Version 2025-11-25, 2025. URL: <https://modelcontextprotocol.io/specification/2025-11-25/server/tools>.
- [34] Model Context Protocol. Server Features: Prompts. Version 2025-11-25, 2025. URL: <https://modelcontextprotocol.io/specification/2025-11-25/server/prompts>.

- [35] Model Context Protocol. Base Protocol: Lifecycle. Version 2025-11-25, 2025. URL:
<https://modelcontextprotocol.io/specification/2025-11-25/basic/lifecycle>.
- [36] Model Context Protocol. Base Protocol: Authorization. Version 2025-11-25, 2025. URL:
<https://modelcontextprotocol.io/specification/2025-11-25/basic/authorization>.