# **Advanced Weed Targeting System**

## **🌱 Project Overview**

The Advanced Weed Targeting System is a real-time, AI-powered precision agriculture solution that combines computer vision, predictive algorithms, and laser control for autonomous weed elimination. The system features advanced trajectory prediction, noise filtering, and autonomous execution capabilities.

### **Key Features**

* **Real-time YOLO-based Weed Detection** - High-accuracy weed identification using YOLOv11n
* **Predictive Trajectory Tracking** - Advanced algorithms predict weed movement with delay compensation
* **Autonomous Laser Targeting** - Automated laser positioning and execution
* **Advanced Noise Filtering** - Multi-stage filtering for stable tracking in field conditions
* **Web-based Control Interface** - Professional dashboard for real-time monitoring and control
* **Multi-platform Support** - Compatible with local cameras and Jetson Nano via MQTT
* **Motor Control Integration** - Precise laser positioning with Helios DAC systems
* **Real-time Parameter Adjustment** - Live tuning of all system parameters

## **🏗️ System Architecture**

┌─────────────────┐ MQTT ┌──────────────────┐ USB/Serial ┌────────────────┐

│ Jetson Nano │◄──────────►│ PC/Server │◄────────────────►│ Laser System │

│ (Optional) │ │ Flask Web App │ │ Helios DAC │

│ - YOLO Model │ │ - Web Interface │ │ Motors │

│ - Camera │ │ - Targeting AI │ └────────────────┘

└─────────────────┘ │ - MQTT Client │

└──────────────────┘

│

▼

┌──────────────────┐

│ Web Browser │

│ Control Panel │

└──────────────────┘

## **🔧 Hardware Requirements**

### **Core Components**

* **Computer/Server**: Windows/Linux PC with Python 3.8+
* **Camera**: USB camera or Jetson Nano with camera module
* **Laser Controller**: Compatible laser system with serial interface
* **Helios DAC** (Optional): For precise motor control
* **Network**: WiFi/Ethernet for MQTT communication

### **Optional Components**

* **Jetson Nano**: For dedicated AI processing
* **Robot Platform**: Mobile base with wheel encoders
* **GPS Module**: For navigation and mapping
* **IMU Sensor**: For heading and acceleration data

## **📋 Software Requirements**

### **Python Dependencies**

bash

Flask==2.3.2

ultralytics==8.0.0

opencv-python==4.8.0.74

numpy==1.24.3

scipy==1.10.1

paho-mqtt==1.6.1

pyserial==3.5

### **Additional Requirements**

* **YOLO Model**: weed3.pt (trained weed detection model)
* **Calibration Data**: calibration\_data.json (camera-laser coordinate mapping)
* **Helios DAC Library**: HeliosLaserDAC.dll (for Windows)
* **⚙️ Configuration**

### **Camera Calibration**

json

{

"calibration\_points": [

{

"camera\_pixel\_x": 640,

"camera\_pixel\_y": 480,

"laser\_x": 2048,

"laser\_y": 2048

}

],

"region\_corners\_camera": [[0,0], [1280,0], [1280,720], [0,720]],

"region\_corners\_laser": [[0,0], [4095,0], [4095,4095], [0,4095]]

}

### **MQTT Topics**

* camera/frame: Base64 encoded camera frames
* camera/detections: Weed detection results
* IMU/data: Robot sensor data
* CAMERA/tracking: Tracking information

## **🎮 Usage Instructions**

### **Starting the System**

1. **Launch the Web Application**

bash

python app.py

1. **Open Web Interface** Navigate to http://localhost:5049 in your browser
2. **Initialize Components**
   * Click "Connect Robot" (if using mobile platform)
   * Click "Connect Laser" to initialize laser system
   * Click "Auto Weed Targeting" to open targeting panel

### **Operating the Targeting System**

#### **Phase 1: System Startup**

1. Click **"Start System"** to initialize YOLO detection
2. Verify camera feed is active
3. Check connection status indicators (green = connected)

#### **Phase 2: Enable Targeting**

1. Click **"Enable Targeting"** to activate autonomous mode
2. System will enter **OBSERVATION** phase
3. Monitor weed detection counts in the panel

#### **Phase 3: Autonomous Operation**

The system operates in three phases:

* **OBSERVATION** (1.0s): Tracks weed movement
* **PREDICTION**: Calculates trajectory and future position
* **EXECUTION**: Autonomous laser targeting and elimination

## **🎛️ Parameter Configuration**

### **Timing Parameters**

* **Observation Time**: Duration to track weed before prediction (0.3-3.0s)
* **Prediction Duration**: How far ahead to predict trajectory (2.0-20.0s)
* **Prediction Delay**: Additional lead time for hardware latency (0.0-5.0s)

### **AI Parameters**

* **YOLO Delay Compensation**: Accounts for AI processing delay (0.5-2.0s)
* **Speed Scaling Factor**: Adjusts predicted movement speed (0.3-1.5)
* **Min Execution Confidence**: Minimum confidence required for targeting (0.1-1.0)

### **Noise Filtering**

* **Filter Strength**: Overall filtering intensity (0.0-1.0)
* **Smoothing Window**: Moving average window size (1-10)
* **Movement Threshold**: Minimum movement to consider real motion (1.0-20.0px)
* **Outlier Threshold**: Maximum allowed position jump (10.0-200.0px)

## **🎯 Control Interfaces**

### **Keyboard Controls**

* **Robot Movement**: W/A/S/D keys
* **Motor Control**: Arrow keys
* **Device Switching**: F1/F2 keys
* **Emergency Stop**: Space bar

### **Web Interface**

* **Real-time Status**: Live system monitoring
* **Parameter Sliders**: Instant parameter adjustment
* **Manual Controls**: Override autonomous operation
* **Data Visualization**: Weed tracking and trajectory display

## **🔧 Advanced Features**

### **Multi-Device Motor Control**

* Support for multiple Helios DAC devices
* Independent control of each laser positioning system
* Real-time position feedback and calibration

### **Trajectory Prediction Algorithms**

* **Fast Trajectory Method**: Uses start-end position for quick response
* **Velocity-based Prediction**: Analyzes movement patterns
* **Consistency Checking**: Validates trajectory accuracy

### **Adaptive Noise Filtering**

* **Outlier Detection**: Identifies and corrects position jumps
* **Moving Average**: Smooths noisy position data
* **Velocity-based Filtering**: Considers movement consistency

## **📊 Monitoring and Diagnostics**

### **Real-time Displays**

* **Weed Count**: Live count of detected weeds by region and movement type
* **System Status**: Current phase and target information
* **Execution Progress**: Active targeting operation status
* **Connection Status**: All hardware component states

### **Performance Metrics**

* **Detection Accuracy**: YOLO confidence scores
* **Tracking Quality**: Trajectory prediction confidence
* **System Latency**: End-to-end response times
* **Execution Success**: Targeting completion rates

## **🐛 Troubleshooting**

### **Common Issues**

#### **No Weed Detection**

* Verify YOLO model file (weed3.pt) exists
* Check camera feed is active
* Ensure lighting conditions are adequate
* Adjust detection confidence threshold

#### **Laser Not Moving**

* Confirm laser controller connection
* Check Helios DAC drivers installed
* Verify calibration data is loaded
* Test motor control manually

#### **MQTT Connection Failed**

* Check broker IP address configuration
* Verify network connectivity
* Ensure MQTT broker is running
* Check firewall settings

#### **Erratic Targeting**

* Increase noise filter strength
* Adjust movement threshold
* Check camera stability
* Verify calibration accuracy

### **Debug Mode**

Enable debug output by setting:

python

self.debug\_mode = True

### **Log Files**

System logs are printed to console with timestamps and component identification.

## **🔌 API Reference**

### **REST Endpoints**

#### **System Control**

* POST /weed-targeting-start: Initialize targeting system
* POST /weed-targeting-stop: Shutdown targeting system
* POST /weed-targeting-toggle: Enable/disable targeting
* GET /weed-targeting-status: Get current system status

#### **Parameter Updates**

* POST /weed-targeting-params: Update system parameters
* POST /weed-targeting-stop-execution: Stop current execution

#### **Hardware Control**

* POST /robot-cmd: Send robot movement commands
* POST /laser-cmd: Control laser operations
* POST /motor-enhanced: Advanced motor control

### **MQTT Messages**

#### **Outgoing**

* Robot control commands
* Laser status updates
* System state changes

#### **Incoming**

* Camera frames (Base64 encoded)
* Sensor data (IMU, GPS)
* Detection results

## **📈 Performance Optimization**

### **Recommended Settings**

* **Observation Time**: 1.0s (balance between speed and accuracy)
* **Filter Strength**: 0.3 (moderate filtering for field conditions)
* **Speed Scaling**: 0.85 (compensates for prediction lag)
* **Min Confidence**: 0.25 (allows targeting of partially occluded weeds)

### **Hardware Optimization**

* Use SSD storage for faster model loading
* Adequate cooling for continuous operation
* Stable power supply for consistent performance
* High-quality camera with good low-light performance

## **🔒 Safety Features**

### **Hardware Safety**

* Emergency stop functionality
* Power level confirmation for high-intensity operations
* Automatic timeout for laser operations
* Safe default positions for all motors

### **Software Safety**

* Parameter validation and bounds checking
* Graceful error handling and recovery
* Connection monitoring and auto-reconnect
* Backup manual control modes

## **📚 Academic References**

This system implements several advanced algorithms:

1. **Kalman Filtering**: For state estimation and noise reduction
2. **Trajectory Prediction**: Based on polynomial fitting and velocity estimation
3. **Computer Vision**: YOLO object detection with custom training
4. **Control Systems**: PID control for precise positioning
5. **Machine Learning**: Deep learning for weed classification