# DESIGN AND IMPLEMENTATION OF PIGS' MOVEMENT INFORMATION TRACKING SYSTEM
/
# 生猪运动信息追踪系统设计与实现

**Jie BAI[1)], Yin HU[4)], Jianhua XUE[2)], Guanzhen LI[6)], Xinyu ZHAO[2)], Wenbao ZHANG[2)], Long WANG[2)], Huabei LI[2)], Zhenyu LIU[4,5)], Linwei LI[*2,3)]**

[1)] Department of Big Data and Intelligent Engineering, Shanxi Institute of Technology, Yangquan 045000, Shanxi/China;
[2)] College of Information Science and Engineering, Shanxi Agricultural University, Taigu 030801, Shanxi/China;
[3)] Key Laboratory of Equipment and Informatization in Environment Controlled Agriculture, Ministry of Agriculture and Rural Affairs, Hangzhou, Zhejiang / China;
[4)] College of Agricultural Engineering, Shanxi Agricultural University, Taigu 030801, Shanxi / China;
[5)] Dryland Farm Machinery Key Technology and Equipment Key Laboratory of Shanxi Province, Taigu 030801, Shanxi / China;
[6)] College of Computer Science and Software Engineering, Hohai University, Nanjin 211100, Jiangsu / China;
*Tel: 0354-6288587; E-mail: sxrllw@126.com*
*Corresponding author: Li Linwei*
*DOI: https://doi.org/10.35633/inmateh-78-11*

***Keywords:*** *Trajectory tracking, motion information, front-back-end separation design architecture, visual object tracking, motion trajectory distribution*

## ABSTRACT
*To address the low efficiency of traditional pig behavior monitoring methods, this study proposes a swine motion information recognition algorithm and develops a corresponding monitoring system. The system adopts a front-end/back-end separated architecture. The front-end provides video playback and control, multi-target identification, and trajectory visualization. The back-end performs motion detection and background modeling using the MOG2 algorithm and generates trajectory heatmaps through DBSCAN-based clustering. Two operational workflows are supported, namely manual annotation and automatic feature extraction. The system calculates key motion parameters, including velocity and momentum, and enables the export of the processed data. Experimental results demonstrate that the proposed system can effectively analyze swine motion characteristics and trajectory information, providing an accurate and efficient monitoring solution for large-scale pig farming, with practical value for optimizing husbandry management and improving animal health.*

## 摘要
*针对传统生猪行为监测效率低的问题，本文设计生猪运动信息识别算法并搭建了对应系统。本文采用前后端分离架构，前端实现视频播控、多目标识别及轨迹绘制，后端利用 MOG2 算法进行运动检测与背景建模，并通过 DBSCAN 算法生成运动轨迹热力图。研究支持基于手动标注与自动提取的双模式识别，可计算速度、动量等运动等参数并导出数据。实验表明，该系统有效分析生猪运动特征和轨迹信息，从而为规模化养殖提供了精准、高效的监测工具，对优化养殖管理和提升动物健康具有实际应用价值。*

## INTRODUCTION
Swine farming represents a critical sector within the livestock industry (*Wang et al., 2024; Mateos et al., 2024*). The rapid shift toward large-scale farming operations has made the use of information technology for intelligent monitoring of pig movement an essential trend to enhance operational efficiency and support management decisions (*Xu et al., 2025; Reza et al., 2024*). As a fundamental tool for behavioral analysis, trajectory tracking provides vital technical support for quantifying movement patterns and assessing health status in pigs (*Huang et al., 2025; Lu et al., 2024; Jaoukaew et al., 2024; Odo et al., 2025; Tu et al., 2024*).

*First author. Mailaddress: 245259485@qq.com. Mailing address: No. 1, Xueyuan Road, Development Zone, Yangquan City, Shanxi Province, China.*
** Corresponding author. Mail address: sxrllw@126.com. Mailing address: No.1, Mingxian South Road, Taigu District, Jinzhong City, Shanxi Province, China. Tel:0354-6288587*

Specifically, trajectory recognition and the analysis of movement parameters are key technologies enabling intelligent monitoring of animal behavior (*Chen et al., 2025; Yu et al., 2025; Tu et al., 2025; Cheng et al., 2025*). By accurately extracting movement trajectories and calculating metrics such as speed and momentum, these methods offer objective insights into individual physiological conditions and group behavioral patterns, thereby laying a foundation for early health warnings and optimized farm management.

Globally, research on intelligent pig monitoring is advancing across multiple dimensions. For instance, *Ocepek et al. (2022)* developed an automated monitoring system based on an improved YOLOv4 model, which optimized tail posture feature extraction and achieved 90% detection accuracy. *Guo et al. (2023)* evaluated three deep learning-based multi-object tracking frameworks and introduced a weighted association strategy to improve re-identification matching, effectively reducing identity switches and enhancing tracking performance on large-scale annotated pig datasets.

However, existing approaches often focus on analyzing single behaviors or individual targets, with limited integration across diverse farming scenarios (*He et al., 2025*). *Cowton et al. (2019)* proposed a deep CNN-based object localization method to build an integrated system capable of autonomously locating and tracking individual pigs, extracting behavior-related metrics from RGB cameras with a mean squared error (MSE) below 0.015 per trajectory. *Liu et al. (2024)* tackled issues of target loss and recognition errors in monitoring drinking behavior by combining LK Pyramid Optical Flow, an improved KCF re-localization method, and DeepLabCut keypoint modeling. Their approach achieved a frame-level duration estimation error below 30 frames, 95% overlap precision (OP), a center location error (CLE) of 3 pixels, and near-real-time processing at approximately 300 fps on 200 video segments. *Zhang et al. (2019)* presented a hybrid method integrating a CNN-based detector with a correlation filter-based tracker. The detector leveraged multi-scale features from different layers within a single-stage prediction network to balance detection accuracy and speed effectively.

It should be noted that current research still lacks capabilities for multi-object collaborative monitoring and demonstrates limited adaptability in resource-constrained environments (*Nidhi et al., 2025*). There is a clear need for the development of lightweight, system-level motion tracking solutions. To address these gaps, this paper proposes a dual-mode motion information recognition system that supports both manual annotation and automatic extraction. The system computes and exports pig motion parameters and trajectory data, offering an integrated solution for smart farming that combines video acquisition, motion analysis, visualization, and data export functionalities.

## MATERIALS AND METHODS

### *System Architecture of Pig Trajectory Tracking System*

#### System Framework Design

This paper presents a web-based system designed with a Browser/Server (B/S) architecture. This design allows users to access the system directly through a web browser without requiring any client-side software installation. Developed using a front-end and back-end separation model, the system enables users to upload videos directly to the server for processing. Communication between the front-end and back-end is handled via Application Programming Interfaces (APIs). The back-end processes the uploaded videos, generating trajectory data and heatmaps, and then sends the resulting data along with visualization files back to the front-end. The front-end renders this information, providing users with video playback and control capabilities. The specific system architecture is illustrated in Fig. 1.

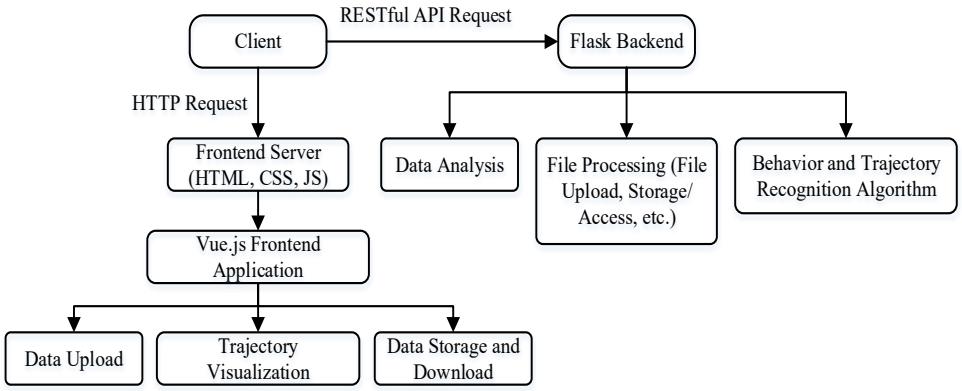



**Fig. 1 - System Architecture Diagram**

On the front-end, the system is built with HTML, CSS, and JavaScript, employing the Vue.js framework for component-based development to enhance both interface structure and code maintainability. Vue.js implements a reactive data-binding mechanism wherein each component instance utilizes a dependency tracker to monitor state properties during rendering. When a tracked property changes, Vue.js automatically triggers re-renders of the dependent components, ensuring the view stays synchronized with the underlying state. This data-driven approach optimizes application responsiveness and developer efficiency.

On the back-end, the lightweight and extensible Flask framework is adopted to construct the service. It handles core business logic, including receiving and managing uploaded video files, invoking algorithm modules for motion analysis, and generating analytical results. RESTful APIs are implemented via Flask's routing mechanism, which maps URL endpoints to specific view functions. Upon receiving an HTTP request, Flask dispatches it to the corresponding function, which executes workflows such as video processing, trajectory and heatmap computation, and parameter calculation. The back-end subsequently returns a structured response—containing status codes, headers, and processed data—to the front-end for display and download.

**Page Design and Layout**

HTML, CSS, and JavaScript constitute the foundation of the system's front-end interface. HTML provides the structural layout of the page, CSS is responsible for visual styling and layout design, and JavaScript handles interactive logic and dynamic state updates. In this work, the aforementioned technologies are employed to implement the interface design and layout of the video playback page, as illustrated in Fig. 2. The playback interface primarily consists of a video display area accompanied by a series of functional controls, including: frame number and timestamp indicators, playback progress and speed adjustment bars, play/pause buttons, configurable start and end position selectors, step size settings, as well as single-step forward/backward navigation and toggles for single/repeat playback modes.

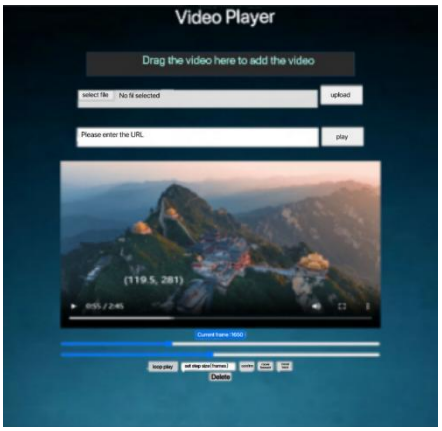



**Fig. 2 - Video Playback Interface**

***Motion Feature Extraction and Implementation***

**Basic Parameter Selection**

To enable quantitative analysis of pigs' actual locomotion in video footage, the system's front-end integrates an interactive parameter calibration module that relies on visual reference scales. A dynamic annotation canvas is rendered in the browser using the Canvas component, with its bottom-left corner serving as the origin of a two-dimensional pixel coordinate system. Once the user clicks to annotate the endpoints of the reference ruler and the target positions on the canvas, the system records the corresponding pixel coordinates and pixel-based distances. These measurements furnish a unified reference for subsequent scale conversion and motion-parameter computations.

To translate pixel measurements into real-world physical scale, the user provides the actual length of the reference ruler. The system then computes the pixel length of that ruler as displayed on the canvas. From these values, the pixel-to-real-length conversion coefficient K is derived, as defined in Equation (1):

$$K = \frac{L_{real}}{L_{pixel}} \tag{1}$$

The actual length input by the user is denoted as $L_{real}$ (in meters), and its corresponding pixel length on the canvas is recorded as $L_{pixel}$ (in pixels).

In addition to scale calibration, the system provides an input interface for pig body weight, allowing kinematic metrics to be extended into dynamic indicators such as momentum and kinetic energy. During data binding, the front-end parses the input value into a numeric type. If the input is empty or cannot be parsed into a valid number, it returns NaN, in which case the system should either skip the calculation of relevant dynamic quantities or prompt the user to re-enter the value. The default unit for mass is kilograms. Fig. 3 illustrates the annotation display after integrating both length and mass parameters.


**Fig. 3 - Ruler with Added Mass and Length Parameters**

Furthermore, to compute the time interval between different video frames, let the frame rate be denoted as $f$ (in frames per second, fps). For two frames with indices $n_i$ (the later frame) and $n_j$ (the earlier frame, where $n_i > n_j$ ), the corresponding time difference is given by Equation (2).

$$\Delta t = \frac{n_i - n_j}{f} \tag{2}$$

This method transforms displacement variations across discrete video frames into continuous time-scale motion characteristics, thereby enabling subsequent analysis and feature extraction of swine locomotor behavior.

**Motion Feature Selection**

Once the scale coefficient $K$ and the time interval $\Delta t$ are obtained, the velocity magnitude (in m/s) of the pig can be calculated based on its real displacement $\Delta s$ between consecutive time points. Here, $\Delta s$ is derived by converting the pixel displacement of trajectory points using the scale coefficient $K$. The velocity is computed as shown in Equation (3):

$$v = \frac{\Delta s}{\Delta t} \tag{3}$$

The acceleration magnitude $a$, which describes the rate of change of the pig's velocity, is obtained from the temporal variation of velocity and is defined as follows:

$$a = \frac{\Delta v}{\Delta t} \tag{4}$$

With the mass $m$ introduced, momentum can be computed to characterize the motion intensity of the pig, expressed in kilogram-meters per second (kg·m/s). The corresponding formula is:

$$p = m \cdot v \tag{5}$$

Kinetic energy is the energy possessed by an object during its motion, and its formula is:

$$E = \frac{1}{2} m \cdot v^2 \tag{6}$$

Simultaneously, vector quantities such as velocity, acceleration, and momentum can be decomposed into X- and Y-axis components to enable directional analysis. In contrast, kinetic energy—a scalar quantity—is computed from the contributions of these velocity components.

To comprehensively analyze the physiological state associated with swine movement postures, this study further incorporates angular and rotational parameters, extending beyond linear motion characteristics. Given the velocity direction angles $\theta_1$ and $\theta_2$ at two consecutive time points, the angular displacement is calculated as follows:

$$\Delta\theta = \theta_2 - \theta_1 \tag{7}$$

Angular velocity describes the rate at which the angle of a pig changes over time, calculated using the formula:

$$\omega = \frac{\Delta\theta}{\Delta t} \tag{8}$$

Here, $\omega$ represents the angular velocity (unit: rad/s), and $\Delta\theta$ is the angular displacement. The angular acceleration $\alpha$—defined as the time derivative of angular velocity—is expressed in rad/s² and given by:

$$\alpha = \frac{\Delta\omega}{\Delta t} = \frac{d^2\theta}{dt^2} \tag{9}$$

The moment of inertia $I$ is a physical property that describes a pig's resistance to changes in rotation about its physiological axis. Its expression is given by:

$$I = mr^2 \tag{10}$$

Here, $I$ denotes the moment of inertia (kg·m²), m is the particle mass (kg), and $r$ is the distance from the particle to the axis of rotation (m).

Torque is the force causing the particle to rotate about its axis, measured in newton-meters (N·m). α is the angular acceleration. The formula is:

$$\tau = I\alpha \tag{11}$$

Angular momentum quantifies the rotational motion of a pig about a given axis and is expressed in kilograms times meters squared per second (kg·m²/s), with ω representing the angular velocity. The corresponding formula is:

$$L = I\omega \tag{12}$$

**Motion Feature Storage**

The system records motion characteristics—velocity and its components, mass, momentum and its components, kinetic energy, angle, angular velocity, angular acceleration, angular displacement, moment of inertia, torque, and angular momentum—into a unified feature array. In the export phase, the front-end coordinates the data based on this array and submits a request to the back-end. Using Python, the back-end writes the data into an Excel worksheet, generates the workbook, and returns the file. The front-end then facilitates browser-based download for local saving, which supports subsequent statistical analysis and long-term retention.

***Trajectory Feature Extraction and Implementation Process***

**Implementation of Manual Position Marking Using Vue.js Front-end**

The manual annotation module supports precise control over annotation positioning and range, facilitating the detailed labeling of selected regions or moving pig targets. This work constructs manual marking and trajectory drawing regions on a front-end canvas and leverages an event-listening mechanism to coordinate interactions between video frames and annotations. Following video import, the system initializes a 2D drawing context on the canvas when the video reaches a playable state. Throughout playback, the current frame is periodically redrawn onto the canvas, ensuring real-time synchronization between the canvas and the video. Users mark pig locations by clicking on the canvas; the system records the canvas coordinates of each click in temporal order. During redraw cycles, the annotation points are connected sequentially and overlaid to render the pig's movement path.

In Vue.js component development, the path-drawing functionality is implemented through coordinated initialization configuration and interaction logic. After loading, the component first acquires the designated canvas element and establishes a 2D drawing environment, thereby providing the underlying interfaces for path rendering. At the user-interaction level, the component encapsulates the logic for adding path points: a canvas click event triggers a response mechanism that captures the coordinates and stores them in a data array. As annotation points are successively added, the coordinates are automatically connected in order to generate a complete movement trajectory.

After completing the path, users can export the drawn trajectory as an image for saving. The flowchart in Fig. 4 illustrates the movement-path drawing implementation.

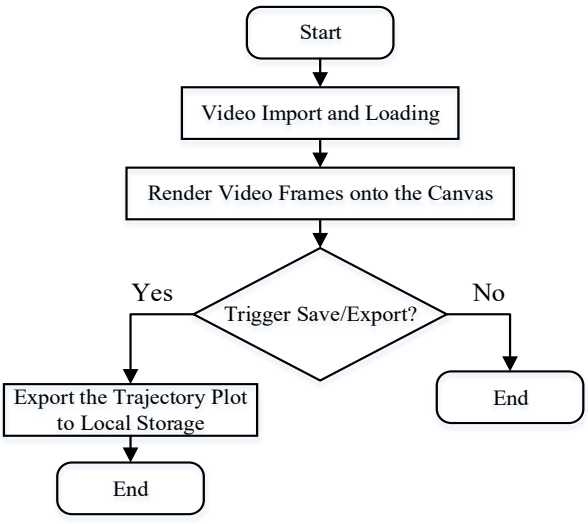



**Fig. 4 - Path Drawing Implementation Flowchart**

To export the path-drawing interface, the canvas content is encoded into a PNG-formatted DataURL string using its built-in image-data extraction capability. A temporary download link is then dynamically created, with the DataURL assigned to its *href* attribute and the output filename specified via the download property. Finally, a click event is programmatically triggered on this link to initiate the automatic download. This approach allows the motion-path diagram to be saved locally under a designated filename, supporting subsequent archiving and analysis.

**Automatic Position Marking Based on the MOG2 Algorithm**

Although manual annotation can achieve high precision, its labor-intensive nature and substantial workload make it unsuitable for large-scale trajectory labeling. To address this limitation, this paper presents an automatic position-marking model based on an improved Mixture of Gaussians (MOG2) algorithm. Compared to the conventional MOG method, the enhanced MOG2 achieves significant improvements in both the efficiency and accuracy of background separation in pig monitoring scenarios, enabling efficient automatic annotation of pig trajectories. The algorithm is built upon a Gaussian Mixture Model (GMM) framework with key optimizations. It dynamically constructs and updates multiple Gaussian distributions to accurately model the complex temporal probability distribution of pixel values (*De et al., 2025*).

In MOG2, the observation $X_t$ of a pixel at time t—which can be a grayscale value or an RGB vector—is modeled by a mixture of K Gaussian distributions. The probability model is as follows:

$$P(X_t) = \sum_{k=1}^{k} w_{k,t} \cdot \eta(X_t; \mu_{k,t}, \sum_{k,t}) \tag{13}$$

Here, $w_{k,t}$ is the weight coefficient for the $k$-th Gaussian distribution at time $t$, $\mu_{k,t}$ is the mean vector, $\sum k,t$ is the variance parameter, and $\eta(\cdot)$ denotes the corresponding Gaussian probability density function.

To determine whether a current pixel matches a particular Gaussian distribution, a variance-normalized matching criterion can be employed:

$$\left|X_t - \mu_{k,t}\right| \leq \lambda \cdot \sqrt{diag(\sum_{k,t})} \tag{14}$$

In the formula, $\lambda$ denotes the adaptive threshold coefficient and $\sqrt{diag(\sum k,t)}$ represents the standard deviation scale. If a pixel matches a particular distribution, that distribution is considered to constitute a valid model for the current observation. The MOG2 algorithm ranks all distributions by their weights and variances, then selects a subset whose cumulative weight meets a predefined background-ratio threshold to form the background model set. A pixel matching this set is classified as background; otherwise, it is designated as a foreground object, thereby generating the foreground mask for pigs.

To accommodate scene changes, the model parameters are updated over time. The weight update is expressed as:

$$w_{k,t} = (1-\alpha) \cdot w_{k,t-1} + \alpha \cdot M_{k.t} \tag{15}$$

In the formula, $\alpha \in (0,1)$ is the learning rate; $M_{k,t}$ is the matching indicator variable, which is 1 for successful matching and 0 otherwise.

For the Gaussian distribution of successful matches, its mean and variance are updated according to the following rules:

$$\mu_{k,t} = (1-\rho) \cdot \mu_{k,t-1} + \rho \cdot X_t \tag{16}$$

$$\sum_{k,t} = (1-\rho) \cdot \sum_{k,t-1} + \rho \cdot (X_t - \mu_{k,t})(X_t - \mu_{k,t})^T \tag{17}$$

Here, $\rho$ serves as an adaptive update coefficient related to the matching degree with the current observation, striking a balance between rapid adaptation and stable modeling. This design prevents the background model from erroneously absorbing foreground pixels during short-term stops or local illumination fluctuations, thereby improving the completeness and stability of the swine silhouette.

Shadow detection is optionally implemented to identify regions with reduced brightness but stable chromaticity, classifying them as shadows rather than foreground. This suppresses false detections from projections and occlusions, improving the reliability of trajectory coordinates.

**Trajectory Extraction and Storage**

The video analysis and processing module takes video file paths as input, performs moving-object extraction and trajectory tracking, and outputs trajectory-overlaid videos along with a folder of results containing trajectories and motion features. The module leverages a video reader to obtain basic video parameters, employs a background subtractor for foreground-background separation, and applies morphological operations for mask refinement. It then locates targets through mask denoising and connectivity restoration, calculates bounding boxes and center points, writes the center-point sequence to a trajectory file, and overlays the trajectory onto the original frames, thereby generating both trajectory visualizations and overlaid video outputs.

The frame-by-frame video processing workflow is shown in Fig. 5, while the target trajectory visualization workflow is depicted in Fig. 6.

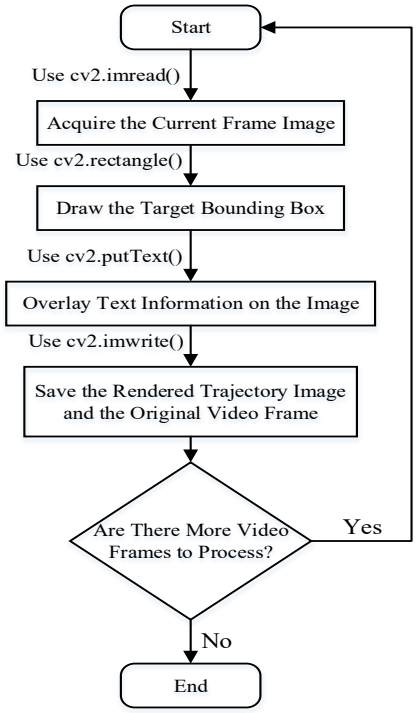



**Fig. 5 - Video Frame-by-Frame Processing Flowchart**

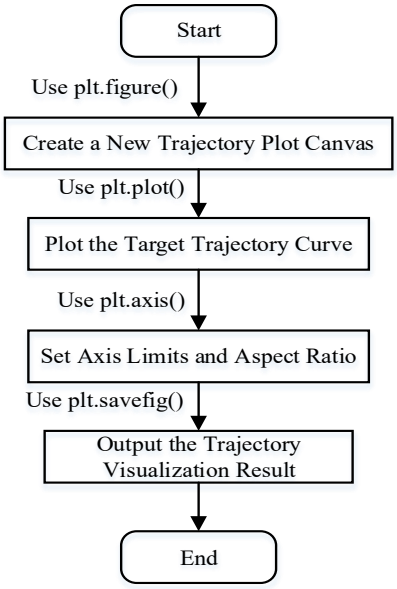



**Fig. 6 - Target Trajectory Visualization Flowchart**

To ensure persistent storage of processed trajectory data and videos, the system creates two directories: "res" for saving video frames with plotted pig trajectories, and "original" for storing the generated trajectory videos together with the original video frames containing motion information. To avoid confusion with previous results, the output directories are cleared at the start of each new task to save storage and facilitate comparison.

The to_video(input_folder) function is used to composite image sequences from a folder into a video file, enabling the conversion of processed frames into a playable video format. The implementation proceeds as follows. First, the output video storage path is set to "static/output/". A unique identifier is generated and appended to the output path to form the complete file path. Next, image files with extensions .jpg or .png are filtered from the target folder and sorted numerically by filename to create an ordered image sequence. The first image is then read to obtain its dimensions as the video resolution; all subsequent frames are validated to maintain consistent size and channel format. Using these parameters, a video writer object is created with the specified video codec, frame rate, and dimensions. The image sequence is then traversed, with each frame read and written to the video file. Finally, the writer is released and the path of the generated video file is returned, completing the conversion from image sequence to video.

***Data Upload and Display***

**Data Upload Implementation**

Within the system, users select pig video files through the front-end interface and trigger an upload request. The video files are received by the back-end built with Flask and stored in a pre-configured upload directory on the server.

To start the service, the back-end is launched by running python app.py in the project directory (after cd videoDisplay). The front-end then accesses the page and performs video upload by connecting to this back-end service address.

The back-end service is constructed using the Flask framework. It defines a route for the home page to handle GET requests and enables Cross-Origin Resource Sharing (CORS) to support cross-domain requests correctly. When a request is received, the system reads the file information from the upload directory and returns a list of uploaded video files in response to the request for the root path, which is then displayed on the front-end. A success message is returned after the operation completes. The detailed workflow for processing GET requests is shown in Fig. 7.

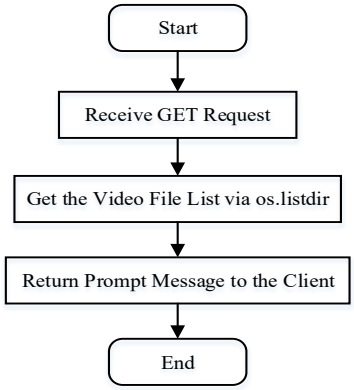



**Fig. 7 - GET Request Flowchart**

To enable file upload, a route is configured to handle POST requests. Within this route, the system extracts the uploaded file from the HTTP request and saves it to a designated storage path. If the file is retrieved and stored successfully, the server returns its access URL to the client for locating the saved file. If no file is detected in the request or if saving fails, a “File not found” error message is returned. The detailed POST request workflow is illustrated in Fig. 8.

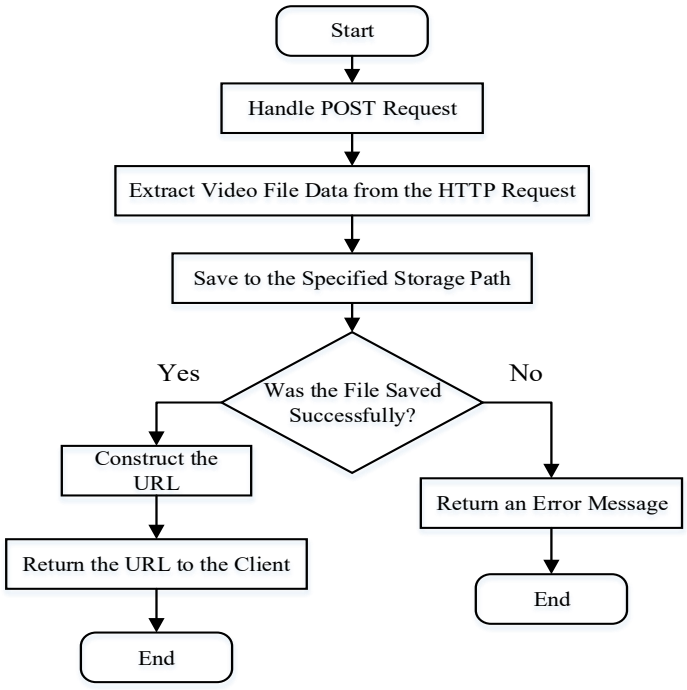



**Fig. 8 - POST Request Flowchart**

### Video Playback and Control Implementation

To enhance the user experience, the system incorporates real-time cursor coordinate display within the interface. At the styling layer, CSS positions and styles the coordinate display element (coordinates) with absolute positioning relative to its parent, ensuring the tooltip follows the cursor smoothly. Text color and font size are adjusted for readability, and mouse events on this element are disabled to prevent interference with video-area interactions.

At the interaction layer, the system listens for *mousemove* events on the video element. When the cursor moves inside the video area, it obtains the cursor's page coordinates and the video element's viewport bounds, then calculates the relative coordinates within the video region. The coordinate display element is updated in real time to track the cursor position, and the computed relative values are rendered on the interface. The implementation result of the cursor coordinate display is shown in Fig. 9.

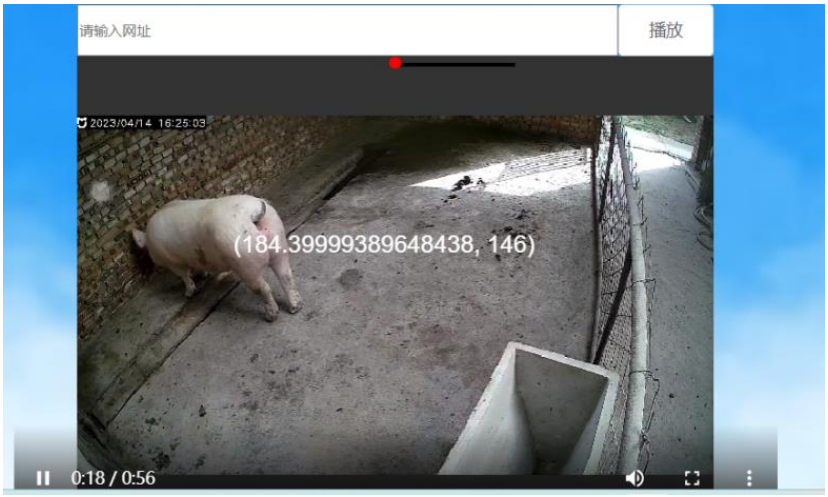



**Fig. 9 - Cursor Coordinate Display**

To enable real-time display of frame count and playback duration, the system includes a frame-display element on the page. This element is bound to the variable frameNumber to dynamically show the current frame number and elapsed playback time. The system listens to the video's time update event, which fires continuously during playback. Inside the event handler, the current playback time currentTime(in seconds) is read, and the current frame number frameNumber is calculated using the preset frame rate frameRate. The result is then written back to the display element, providing real-time synchronization of the frame count and time. The workflow of this feature is illustrated in Fig. 10.

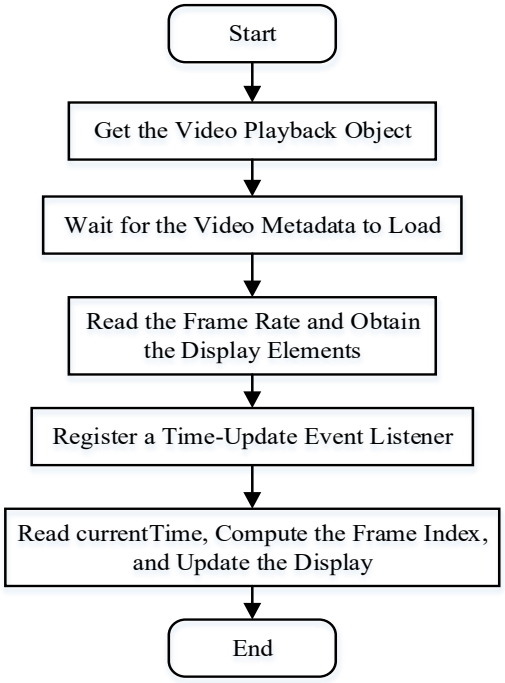



**Fig. 10 - Real-Time Viewing Function Flowchart**

### *Pig Movement Tracking and Mapping*

#### Manual/Automatic Dual Mode Switching

To balance the flexibility of trajectory acquisition with processing efficiency, the system incorporates two operational modes, manual trajectory annotation and automatic trajectory tracking, with seamless switching between modes enabled through front-end routing.

The manual mode uses Home.vue as the default entry point. After loading a video in the playback interface, users can perform point-by-point annotations on the front-end canvas overlay. The system records the two-dimensional pixel coordinates corresponding to each user click in chronological order, and consecutive points are connected to form trajectory curves for visualization. The recorded coordinate sequences can subsequently be used for motion parameter computation and result export.

The automatic mode is triggered via the automatic trajectory entry in Home.vue, which initiates route navigation to the auto page. The automatic tracking interface provides a return mechanism implemented in App.vue, allowing users to navigate back to the manual interface and thereby enabling bidirectional switching between manual and automatic modes.

To support automatic tracking, App.vue integrates a video file upload module. Upon selecting and submitting a video, the front-end encapsulates the file as form data and structures it into a request payload, which is then transmitted to a designated local server endpoint. Upon successful upload, the back-end returns a video access URL for subsequent visualization or processing. In the event of upload failure, an error message is returned. The video upload workflow for the automatic tracking interface is illustrated in Fig. 11.

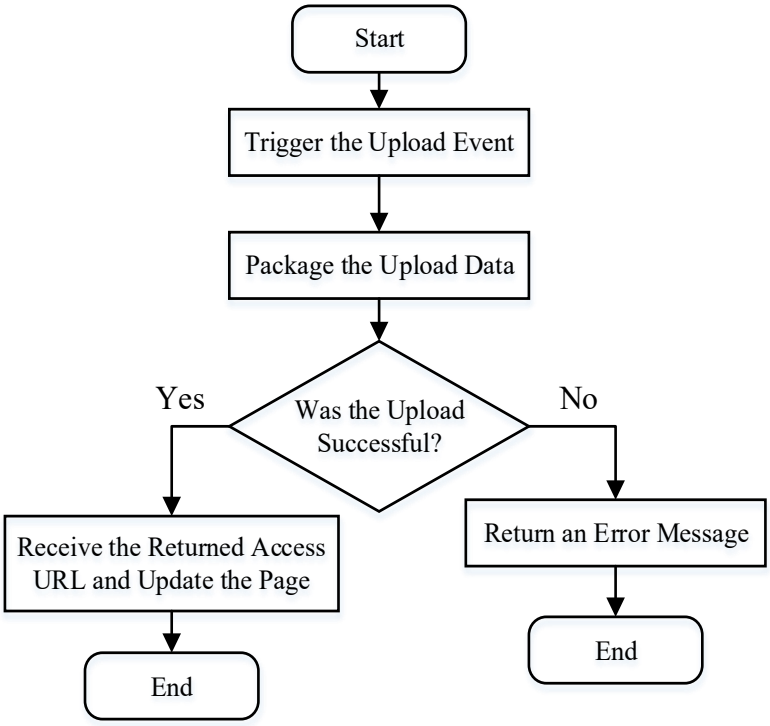



**Fig. 11 - Video Upload Flowchart for the Auto-Tracking Interface**

**Parameter and Path Display**

To meet the requirements for multi-target tracking and annotation, the system adopts a channel-based interactive design featuring three independent control modules. Users can establish parallel data channels by selecting “Add Marker 1–3,” with each channel corresponding to the annotation point sequence of a specific target. Based on real-time coordinate feedback, data within a channel can be either retained or reset. Coupled with the “Add Data 1–3” function, this design enables targeted data entry and management, resulting in a structured approach to data organization. This channel-separated mechanism for data collection and quality control reduces interaction conflicts during simultaneous multi-target annotation and helps prevent data contamination caused by operational errors.

To facilitate users in intuitively understanding the motion state of Duroc pigs, the system overlays the movement trajectory on the video frame. The system synchronously annotates the speed and the speed components along the X direction and the Y direction near the trajectory.

The result is shown in Fig. 12. The example in Fig. 12 was collected in an environment with a length of 9 m and a width of 1.5 m. The subject was a 6-month-old Duroc pig.

**Fig. 12 - Multi-target Punching Path Diagram**

Taking the example illustrated in Fig.12, the system outputs multiple kinematic and dynamic metrics in real time during trajectory visualization, allowing users to observe movement trends of the pigs. For linear motion, these metrics include displacement and velocity—along with their X and Y components—as well as acceleration, momentum, and kinetic energy, also resolved into directional components where applicable. Together, these measurements characterize movement intensity from the perspectives of velocity change and energy expenditure.

After background modeling and foreground separation of the video using the MOG2 algorithm, the back-end extracts the trajectory point sequence in Fig. 12 based on the target region localization results, and plots the motion trajectory accordingly. The result is shown in Fig. 13. After the user completes video upload and confirms execution, the back-end automatically processes the video frame by frame and outputs the corresponding trajectory visualization result.

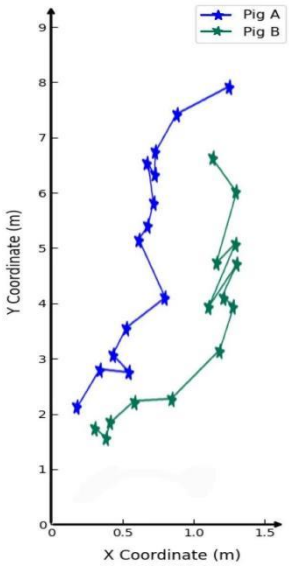



**Fig. 13 - Mapping the movement trajectories of pigs**

The distribution of pig dwell times across different spatial zones reflects their behavioral preferences and environmental adaptation. However, when long-term trajectories are overlaid on the same frame, significant overlap often obscures the identification of high-activity areas. To address this, the present study employs the DBSCAN (Density-Based Spatial Clustering of Applications with Noise) algorithm to cluster and visualize dwell hotspots, thereby improving the interpretability of long-term movement density.

In practice, the system integrates the MOG2 algorithm within a Flask-based back-end to process video frames sequentially and record trajectory coordinates. Trajectories are then mapped based on the sequence of points, and the resulting trajectory images are exported chronologically to a specified directory. To avoid interference from previous results, the system automatically clears this directory before processing each new video task. When dealing with videos containing a large number of frames, the frame-extraction strategy can be adjusted in the back-end to balance processing efficiency and trajectory accuracy.

During heatmap generation, the system processes each trajectory image through grayscale conversion and binarization to minimize color interference. Contour points of the trajectory are then extracted from the binary images and organized into NumPy arrays as input samples for clustering. DBSCAN is applied to spatially cluster these trajectory points and assign a cluster label to each one. These labels are combined with the coordinate data to form a two-dimensional feature set containing categorical information. Using Matplotlib's density mapping functionality, the clustering results are converted into a color-temperature distribution map, where varying color intensities visually represent the spatial density of trajectory points, thereby revealing regional patterns of pig activity hotspots.

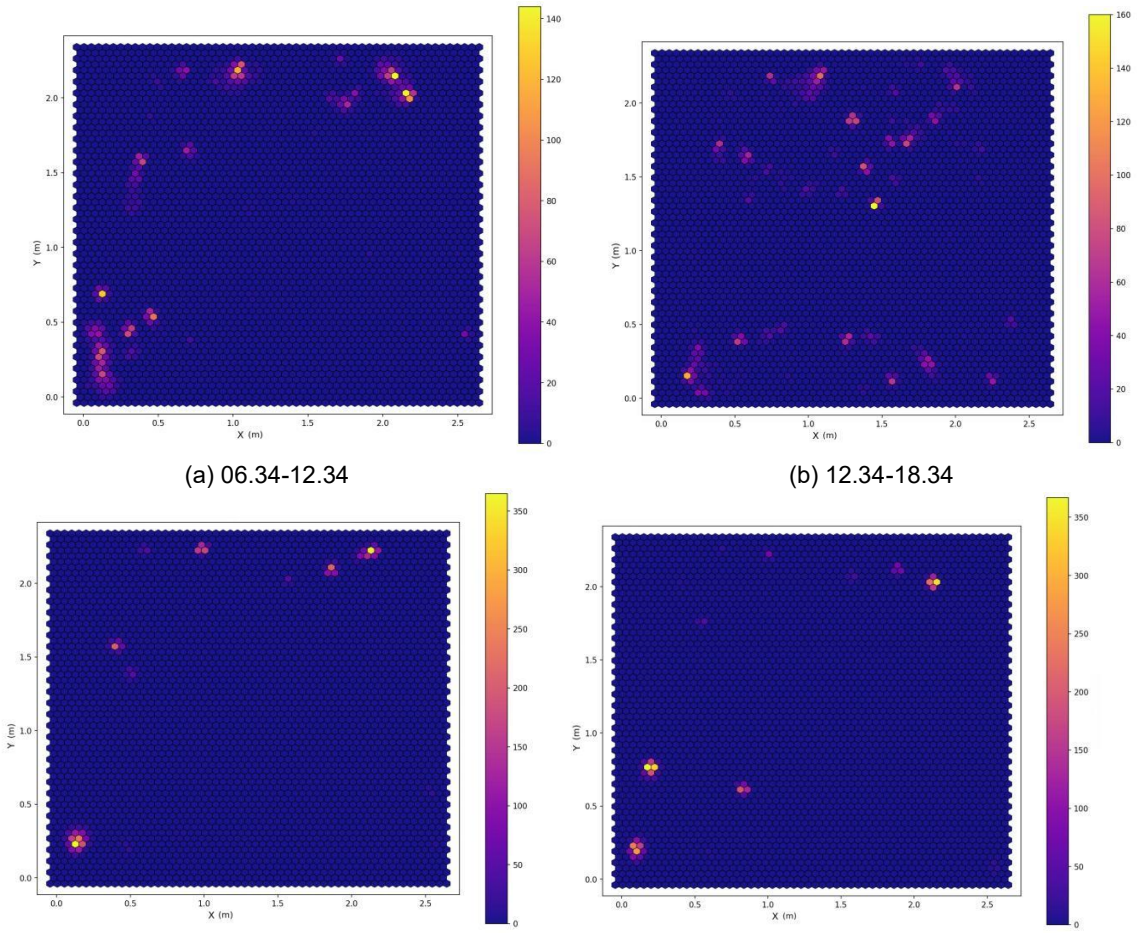
## RESULTS

### *Test Results and Analysis*

Experimental data were collected from pigpen surveillance videos. The acquisition device was a Hikvision C3W intercom camera with a 2.8 mm focal length and 4 MP resolution. The experiment was conducted in a pen environment with a length of 2.6 m and a width of 2.3 m. A 10-week-old Duroc pig was recorded continuously for 24 h. To analyze daily activity patterns, the 24 h video was divided into four consecutive 6 h periods, including 06:34 to 12:34, 12:34 to 18:34, 18:34 to 00:34, and 00:34 to 06:34 on the next day. These periods correspond to real clock time during recording. Recording started at 06:34 on the recording day. The last period crossed to the next day. Results in this chapter are consistent with the Materials and Methods in the previous chapter. Trajectory point extraction used MOG2-based background modeling and foreground separation to obtain a foreground mask. Morphological denoising and connected component analysis were applied to locate target regions. Target center points were then computed to form a frame-by-frame trajectory sequence. Coordinate representation followed the scale calibration and coordinate mapping method in the previous chapter. Pixel coordinates were converted to real-scale coordinates, and density statistics were completed.

Fig. 14 shows differences in pig activity spatial distribution across different time periods. Subfigures a to d are four subfigure indices and correspond to four consecutive 6 h periods. The time label under each subfigure indicates the data range used for statistics. For example, 06:34 to 12:34 indicates statistics computed only from all trajectory points extracted in this period. For clear interpretation of spatial positions and numerical meaning, Fig. 14 uses the pen planar coordinate system for presentation. The lower left corner of the pen is the coordinate origin. The X axis represents the pen length direction with a range from 0 to 2.6 m. The Y axis represents the pen width direction with a range from 0 to 2.3 m. The coordinate unit is meters. The heatmap color represents activity frequency within each spatial grid cell. The color bar provides the corresponding numerical scale. A larger value indicates more visits or longer stays in the region during the period.

At long time scales, line-based trajectory plots are crowded due to trajectory overlap, and key regions are difficult to locate quickly. The heatmap applies gridded statistics to trajectory points and summarizes movement into a spatial distribution of visit frequency or dwelling intensity. The heatmap highlights high-activity hotspots and supports intuitive comparison and quantitative analysis across different time periods. Fig. 14 shows clear differences in spatial density across different periods. During 06:34 to 12:34, hotspots were mainly concentrated in local regions of the pen. Only a few locations showed high-frequency visits and stays. This pattern indicates a relatively localized activity space and stable dwelling regions. These regions are usually associated with repeated behaviors such as feeding, drinking, or short rest. During 12:34 to 18:34, hotspots were more dispersed and covered a larger area. This pattern indicates an expanded activity range, more frequent cross-region movement, and higher spatial utilization. During 18:34 to 00:34, overall hotspot intensity decreased and only a few local high-value points remained. This pattern indicates reduced activity intensity with more short-distance movement and local stays. During 00:34 to 06:34 on the next day, hotspots further decreased and showed more stable clustered positions. This pattern reflects night-time resting and dwelling behavior as the main state, and cross-region movement was significantly reduced. Overall, time-segmented heatmaps can directly reveal daily changes in activity intensity and major dwelling regions of pigs, and the results provide spatial distribution evidence for movement monitoring and feeding management.




(a) 06.34-12.34 (b) 12.34-18.34

(c) 18.34-0.34 (d) 0:34–6:34 the following day

**Fig. 14 - Heatmap of Hog Movement Trajectories Across Different Time Periods**

## CONCLUSIONS

To address the inefficiencies of traditional pig behavior monitoring—such as the lack of continuous tracking and multi-target collaborative observation—this study designed and implemented a pig movement tracking system based on Vue.js and Flask. The system adopts a B/S architecture, allowing direct browser access, which significantly improves deployability and usability.

Functionally, the front-end, built with Vue.js, supports video playback and control, multi-target trajectory plotting, and real-time cursor coordinate display. It uses data-driven synchronization to update interface states and results dynamically. The back-end, developed with the Flask framework, handles business logic in Python. It employs MOG2 background modeling to separate foreground objects and extract trajectory points of moving targets. Combined with DBSCAN density clustering, the system identifies and visualizes trajectory hotspots, generating heatmaps that intuitively display activity density distributions. The system supports both manual annotation and automatic extraction for trajectory acquisition, calculates and exports kinematic metrics from trajectory sequences, and enables standardized storage and visualization of trajectory data.

Experimental and application results show that the system can effectively extract, display, and statistically analyze pig movement trajectories from conventional surveillance videos. It enhances the automation and visualization of behavioral monitoring, offering technical support for daily management and health assessment in large-scale farming. Future work may focus on improving robustness under complex occlusion and varying lighting conditions, expanding behavioral metrics, and enhancing multi-terminal compatibility to broaden the system's practical applicability.

## ACKNOWLEDGEMENT

The authors would like to thank all the funding projects for their support in this study:

- Project No. CXGC2025076, "Scientific and Technological Innovation Enhancement Project" of Shanxi Agricultural University.
- Project No. 2011NYZD2202, Open Program Foundation of the Key Laboratory of Equipment and Informatization in Environment Controlled Agriculture, Ministry of Agriculture and Rural Affairs of China.

- Project No. 202302010101002, Key R&D Program of Shanxi Province.
- Project No. 2023-092, Shanxi Scholarship Council of China.

## REFERENCES

[1] Chen, J., Liu, L., Li, P., Yao, W., Shen, M., Ding, Q. A., & Liu, L. (2025). PKL-Track: A Keypoint-Optimized approach for piglet tracking and activity measurement. *Computers and Electronics in Agriculture*, 237, 110578.

[2] Cheng, T., Sun, F., Mao, L., Ou, H., Tu, S., Yuan, F., & Yang, H. (2025). The group-housed pigs attacking and daily behaviors detection and tracking based on improved YOLOv5s and DeepSORT. *PLoS One*, 20(10), e0334783.

[3] Cowton, J., Kyriazakis, I., & Bacardit, J. (2019). Automated individual pig localisation, tracking and behaviour metric extraction using deep learning. *IEEE Access*, 7, 108049-108060.

[4] De Castro, A. L., Wang, J., Bonney-King, J. G., Morota, G., Miller-Cushon, E. K., & Yu, H. (2025). AnimalMotionViz: An interactive software tool for tracking and visualizing animal motion patterns using computer vision. *JDS communications*, 6(3), 416-421.

[5] Guo, Q., Sun, Y., Orsini, C., Bolhuis, J. E., de Vlieg, J., Bijma, P., & de With, P.H. (2023). Enhanced camera-based individual pig detection and tracking for smart pig farms. *Computers and Electronics in Agriculture*, 211, 108009.

[6] He, Z., Nidhi, M. H., Guo, Z., Lyu, L., Guo, C., Hou, J., Wang, X., & Liu, K. (2025). Harnessing contactless monitoring technology for sows and piglets within farrowing pens: a critical review. *Smart Agricultural Technology*, 101321.

[7] Huang, Y., Liu, K., Lv, Y., Xiao, D., Liu, J., & Tan, Z. (2025). Behavior tracking and analysis of group-housed pigs based on deep OC-SORT. *Computers and Electronics in Agriculture*, 239, 111070.

[8] Jaoukaew, A., Suwansantisuk, W., & Kumhom, P. (2024). Robust individual pig tracking. International *Journal of Electrical and Computer Engineering (IJECE)*, 14(1), 279-293.

[9] Liu, C., Ye, H., Wang, L., Lu, S., & Li, L. (2024). Novel tracking method for the drinking behavior trajectory of pigs. *International Journal of Agricultural and Biological Engineering*, 16(6), 67-76.

[10] Lu, J., Chen, Z., Li, X., Fu, Y., Xiong, X., Liu, X., & Wang, H. (2024). ORP-Byte: A multi-object tracking method of pigs that combines Oriented RepPoints and improved Byte. *Computers and Electronics in Agriculture*, 219, 108782.

[11] Mateos, G.G., Corrales, N.L., Talegón, G., Aguirre, L. (2024). Invited Review. Pig meat production in the European Union-27: current status, challenges, and future trends. *Animal bioscience*, 37(4), 755–774.

[12] Nidhi, M. H., Liu, K., & Flay, K. J. (2025). Multi object tracking in livestock-from farm animal management to state-of-the-art methods. *Animal*, 101503.

[13] Ocepek, M., Žnidar, A., Lavrič, M., Škorjanc, D., & Andersen, I. L. (2022). DigiPig: First Developments of an Automated Monitoring System for Body, Head and Tail Detection in Intensive Pig Farming. *Agriculture*, 12(1), 2.

[14] Odo, A., Mclaughlin, N., & Kyriazakis, I. (2025). Re-identification for long-term tracking and management of health and welfare challenges in pigs. *Biosystems Engineering*, 251, 89-100.

[15] Reza, M. N., Ali, M. R., Kabir, M. S. N., Karim, M. R., Ahmed, S., Kyoung, H., Kim, G., & Chung, S. O. (2024). Thermal imaging and computer vision technologies for the enhancement of pig husbandry: a review. *Journal of Animal Science and Technology*, 66(1), 31.

[16] Tu, S., Ou, H., Mao, L., Du, J., Cao, Y., & Chen, W. (2024). Behavior Tracking and Analyses of Group-Housed Pigs Based on Improved ByteTrack. *Animals*, 14(22), 3299.

[17] Tu, S., Ou, H., Yang, A., Liang, Y., Du, J., Cao, Y., He, R., & Yuan, F. (2025). Pig Aggression Tracking and Analysis Application Based on A RPMeMOTR Method. *Smart Agricultural Technology*, 101311.

[18] Wang, L., & Li, D. (2024). - Invited Review - Current status, challenges and prospects for pig production in Asia. *Animal bioscience*, 37(4), 742–754.

[19] Xu, J., Ying, Y., Wu, D., Hu, Y., & Cui, D. (2025). Recent advances in pig behavior detection based on information perception technology. Computers and Electronics in Agriculture, 235, 110327.

[20] Yu, S., Baek, H., Son, S., Seo, J., & Chung, Y. (2025). FTO-SORT: a fast track-id optimizer for enhanced multi-object tracking with SORT in unseen pig farm environments. *Computers and Electronics in Agriculture*, 237, 110540.